Technická univerzita v Liberci
Fakulta strojní

DIPLOMOVÁ PRÁCE

Laboratorní model mikropočítačového řízení portálového jeřábu

Liberec 2005 Michal Špiryt

Technická univerzita v Liberci
Fakulta strojní

Studijní obor : 23 – 40 – 8 Automatizované systémy řízení ve strojírenství
Zaměření : Automatizace inženýrských prací

Katedra aplikované kybernetiky

Laboratorní model mikropočítačového řízení portálového jeřábu

Laboratory model of microprocessor controlling of the bridge crane

Michal Špiryt

Vedoucí diplomové práce : Ing. Slavomír Němeček
Konzultant diplomové práce: Miloš Hejzlar

ANOTACE

Technická univerzita v Liberci
Fakulta strojní

Katedra aplikované kybernetiky

Studijní obor :	23 – 40 – 8 Automatizované systémy řízení ve strojírenství
Studijní zaměření :	Automatizace inženýrských prací
Diplomant :	Michal Špiryt
Téma práce :	Laboratorní model mikropočítačového řízení portálového jeřábu
Theme of work :	Laboratory model of microprocessor controlling of the bridge crane
Rok obhajoby DP :	2005
Vedoucí DP :	Ing. Slavomír Němeček
Konzultant DP :	Miloš Hejzlar

Stručný výtah :

Cílem této diplomové práce je návrh a realizace laboratorního modelu portálového jeřábu, který by měl rozšířit laboratoř katedry aplikované kybernetiky fakulty strojní TUL.

Model je realizován jako portálový jeřáb přepravující materiál z jednoho místa na druhé. Celý model se skládá z vozidla, portálového jeřábu a PC, které pracuje jako nadřazený systém. Tento model by měl umožnit studentům získat základní znalosti o práci mikrokontrolérů a jejich použití při řízení servomechanismů.

Abstract:

The aim of this thesis is a design and a implementation of a laboratory model of a bridge crane which should extend the laboratory of the Department of Applied Cybernetics at the Mechanic Faculty TUL.

The model is implemented as the bridge crane transporting material from one point to another. The model itself includes the vehicle, the bridge crane and the PC that works as the superior system. This model is supposed to enable students to gain a basic knowledge of microcontrollers work and their use in servomechanism controlling.

Poděkování

Na tomto místě je mojí milou povinností poděkovat Ing. Slavomíru Němečkovi a Miloši Hejzlarovi za odborné vedení, cenné rady, poskytnuté informace a za pomoc při zpracování diplomové práce.

Prohlášení o využití diplomové práce

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/200 Sb. O právu autorském, zejména §60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

V Liberci 28.5.2008

……

Michal Špiryt

Místopřísežné prohlášení

„Místopřísežně prohlašuji, že jsem diplomovou práci vypracoval samostatně s použitím uvedené literatury.“

V Liberci 28.5.2005

……

Michal Špiryt

Obsah

1. Úvod

Hlavním cílem této diplomové práce je vytvoření didaktické pomůcky pro studenty, kteří se zabývají předměty Servomechanismy a Číslicové počítače na katedře aplikované kybernetiky. Touto pomůckou má být model systému pro přepravu materiálu. Tento model se skládá z modelu portálového jeřábu a modelu autonomního vozíku, které spolu navzájem musí komunikovat.

Nejdůležitější je ukázat studentům možnosti jednočipových mikropočítačů při regulaci a možnost ovládání mikrokontrolérů přes osobní počítač. Každý, kdo s tímto modelem bude pracovat, by měl mít základní vědomosti v oblasti elektroniky, automatizovaného řízení a základní znalosti v programování mikrokontrolérů.

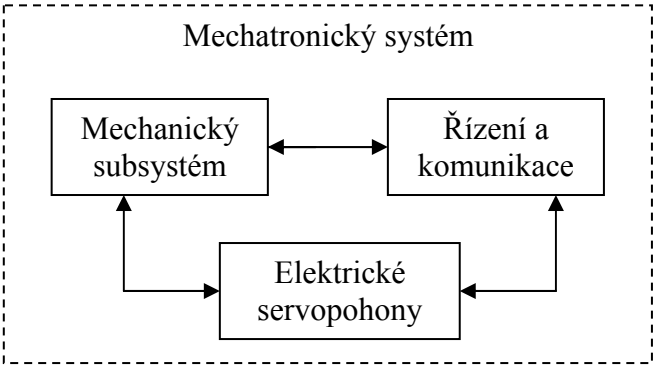
Při realizaci úlohy budou řešeny především otázky spojené s regulací polohového servomechanismu. Protože bude mít jeřáb pro hlavní podélný pohyb k dispozici dva nezávislé motory, musí se také navrhnout řízení těchto dvou motorů tak, aby nedocházelo ke zpříčení celého jeřábu například z důvodu prokluzu poháněného kola. Součástí řešení budou i přehledná schémata zapojení mikrokontroléru a okomentované části zdrojového textu, které pomohou každému čtenáři pochopit chování celého modelu. Některé části tohoto zdrojového kódu bude možné použít i při řešení jiných problémů týkajících se servomechanismů řízených pomocí jednočipových mikropočítačů, protože i ty nejsložitější části programu se skládají z menších funkčních celků.

2. Servopohony

2.1 Elektrický servopohon jako systém

Elektrický servopohon v mechatronice představuje subsystém mechatronického systému, v němž zajišťuje řízení pohybu při současné přeměně elektrické energie v mechanickou práci. Dalšími subsystémy mechatronického systému jsou subsystém řízení a komunikace a subsystém mechaniky (*Obr.2.1*).

Elektrický servopohon je regulační pohon, který se skládá z jednoho nebo více elektrických motorů, napájecích výkonových měničů a řídících a regulačních obvodů. Zatímco běžný elektrický pohon může pracovat v řadě případů i v otevřené regulační smyčce (bez zpětné vazby), servopohon je zapojen vždy v uzavřené regulační smyčce, se zpětnou vazbou proudovou**,** rychlostní popř. polohovou a většinou i se všemi třemi najednou. Servopohon lze rozdělit na dva subsystémy, a to na výkonový subsystém a subsystém informační. Do okolí servopohonů se potom zařazují : elektrická napájecí síť, mechanický subsystém (poháněné pracovní zařízení), pracovní prostředí a v neposlední řadě také subsystém řízení. Výkonový subsystém se skládá z elektrických servomotorů, napájecích výkonových měničů a jiného příslušenství. Informační subsystém je pak tvořen z řídící elektroniky a regulačních obvodů.



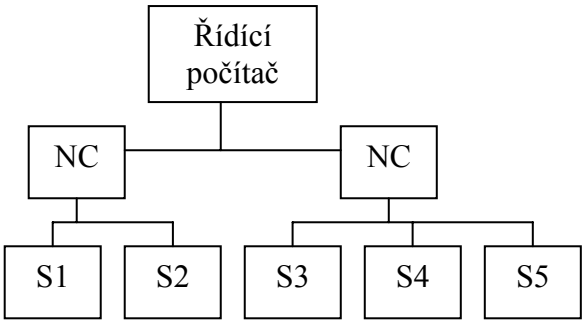
Obr.2.1 Subsystémy mechatronického systému

[1, 2]

2.2 Systémy řízení servopohonů

Nejjednodušší autonomní servopohony individuálně využívaných pracovních strojů mohou být ovládány ručně z ovládacího panelu stroje. Častější je však případ, kdy servopohony jsou součástí rozsáhlejšího systému řízení, nejčastěji uspořádaným hierarchicky. Na nejnižší úrovni (což znamená nejblíže technologickému procesu) jsou servopohony (autonomní rychlostní nebo polohové servomechanismy). Tato úroveň řízení zajišťuje dynamiku pohybu a zpětná diagnostická hlášení. Střední úrovní řízení je

nejčastěji řídící počítač (numerický řídící systém), který řídí v reálném čase technologický proces (polohu, rychlost pohybu, kroutící moment nebo sílu). Řídící počítače jednotlivých pracovních strojů jsou zapojeny např. do lokální počítačové sítě a řízeny z nadřazeného počítače. Hierarchický systém řízení vždy předpokládá, že při poruše vyšší úrovně řízení je nižší úroveň schopna autonomní funkce s určitým omezením. Schéma hierarchického uspořádání je naznačeno na *Obr.2.2.*.



Obr.2.2 Schéma hierarchického řízení

[1, 2]

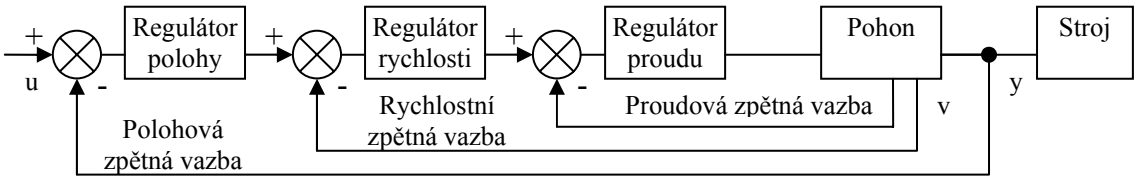
2.3 Servopohon jako polohový servomechanismus

Elektrický servopohon je regulační pohon, který se skládá z elektromotoru, polovodičového měniče pro napájení a řízení motoru a z regulátoru otáček, respektive polohy. Součástí motoru jsou obvykle snímače rychlosti a polohy. Servopohon pracuje v uzavřené zpětné vazbě.

Rychlostní servopohon má pouze rychlostní zpětnou vazbu a umožňuje rychlé a přesné sledování zadávané rychlosti. Polohový servopohon je servomechanismus pro řízení polohy a to buď prostřednictvím převodu posuvné dráhy nebo úhlu natočení. Z hlediska aplikace se rozlišují dva typy polohových regulací, a to cílová a sledovací. Cílová regulace slouží k časově optimálnímu nastavování polohy (např. manipulátory). Časově optimální nastavení polohy představuje polohové přemístění v nejkratším možném čase bez překmitu. Sledovací polohová regulace se používá v případě, že je potřeba sledovat zadanou trajektorii (např. roboty).

Přesné řízení polohy vyžaduje uzavřenou polohovou vazbu, jejíž princip spočívá v porovnání zadané polohy se skutečnou polohou, která se získá z odměřování. Pro sledovací servomechanismy i servomechanismy s cílenou regulací se nejčastěji používá kaskádní struktura regulačních smyček, v níž je polohová smyčka nadřazená vnitřní rychlostní smyčce. K vyhodnocení odchylky mezi žádanou a skutečnou polohou se pak používá proporcionálních regulátorů polohy. Jako další podřízená smyčka rychlostní smyčky může být smyčka proudová. Tato smyčka nejnižší úrovně – proudová – slouží

k ochraně motoru proti přetížení a k omezení maximálního momentu, kterým může působit motor na mechanickou soustavu. U sevomechanismů rychlostních a polohových, napájených pomocí tranzistorových měničů, kompenzuje vlivy indukčnosti kotvy servomotorů, indukovaného napětí a umožňuje prudký nárůst proudu kotvy při reakcích servomotoru na změny vstupních signálů (poruchy, žádaná hodnota reg. veličiny apod.). Při použití tranzistorových měničů s pulzní šířkovou modulací je proudová smyčka velmi rychlá, což je dobrý předpoklad kvality nadřazené rychlostní smyčky. Princip polohového servomachanismu je patrný z *Obr.2.3*.



Obr.2.3 Blokové schéma servomechanismu

[1, 2, 6]

2.4 Způsoby odměřování a řízení polohy

K odměřování skutečné polohy slouží snímače polohy. Podle použitého principu odměřování to jsou snímače absolutní, inkrementální a cyklicky absolutní.

Absolutní snímače udávají absolutní hodnotu polohy v celém rozsahu odměřování, tzn. každé poloze je přiřazena jednoznačně hodnota signálu snímače a naopak. Předností absolutního odměřování je existence informace skutečné poloze hned po zapnutí zařízení. Jako absolutní snímače se používají např. snímače potenciometrické, selsyny, optické s kódovacími kotoučky a jiné.

Inkrementální snímače mají vysokou přesnost i rozlišitelnost, avšak informaci o absolutní poloze lze získat pouze tak, že po zapnutí servomechanismus najede na tzv. referenční bod (nulovou polohu) v dané ose. Skutečná absolutní poloha je pak dána obsahem čítače odměřovacích impulzů. Z konstrukčního hlediska mohou být inkrementální snímače jak rotační, tak lineární.

Cyklicky absolutní snímače odměřují absolutní polohu pouze v omezené oblasti (např. v rozsahu jedné otáčky motoru). Údaj o absolutní poloze je opět pouze uložen v paměti řídícího systému. Cyklicky absolutní snímače mohou být jak rotační (selsyny, resolvery), tak lineární (induktosyny).

Podle umístění snímače polohy rozlišujeme tzv. přímé a nepřímé odměřování. Při nepřímém odměřování je rotační snímač umístěn na hřídeli motoru. Výhoda nepřímého odměřování spočívá v tom, že nelinearity mechanických převodů jsou vně uzavřené polohové smyčky. Ovšem tyto nelinearity mechanického převodu způsobují dodatečnou chybu řízení, kterou již nelze regulačně ovlivnit. Případné chyby je však možné řešit kompenzací parazitních nelinearit nebo řešit celý systém jako nelineární, a to v případě, že jsou k dispozici charakteristiky nelinearit (např. viskózní tření, vůle v zubech apod.).

Při přímém odměřování se používá buď lineárních snímačů polohy, je-li výsledný pohyb posuvný, nebo snímačů rotačním, je-li výsledný pohyb rotační. Nelinearity mechanického převodu jsou v tomto případě uvnitř uzavřené polohové smyčky. Přesnost řízení polohy je vyšší ve srovnání s nepřímým odměřováním.

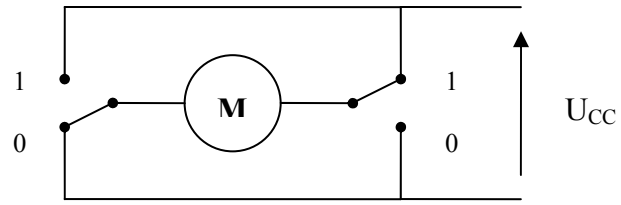
[1, 2, 3]

2.5 Servopohony se stejnosměrnými motory

Stejnosměrný motor s konstantním magnetickým tokem, tzn. motor s cizím buzením nebo permanentními magnety, je ideálním regulačním motorem, protože jeho otáčky lze plynule měnit pomocí změny přiváděného napětí.

Pro servopohony se používá zejména stejnosměrných motorů s cizím buzením permanentními magnety ve statoru. Pro větší výkony se používají motory s cizím buzením, v nichž se magnetické pole vytváří proudem budícího vinutí, navinutém na tzv. hlavních pólech statoru. Jako napájecí obvody těchto motorů se pak používají buď tyristorové měniče nebo měniče tranzistorové, jejichž základním stavebním kamenem je výkonový tranzistor nejčastěji v provedení NPN, který je kvůli velkému proudovému zesílení zapojen se společným □ditorem.

Tranzistorový měnič pro stejnosměrný motor se skládá zpravidla ze čtyř spínačů zapojených v můstkovém zapojení. Princip činnosti je patrný ze zjednodušeného zapojení s kontakty podle *Obr2.4.*



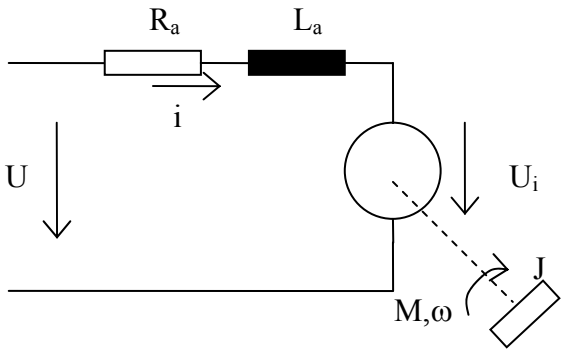
Obr 2.4 Náhradní schéma tranzistorového měniče

2.5.1 Matematický model stejnosměrného motoru

Stejnosměrný motor jako dynamický systém je popsán soustavou diferenciálních rovnic, které je možno odvodit z náhradního schématu (viz *Obr.2.5*), zahrnujícího i indukčnost vinutí kotvy.

$$U = R_a i + L_a \frac{di}{dt} + C\Phi\omega \tag{2.1}$$

$$C\Phi i = J\frac{d\omega}{dt} + B\omega + M_0 \tag{2.2}$$



Obr.2.5 Schéma stejnosměrného motoru

,kde C je konstanta motoru (napěťová, momentová), Φ je magnetický tok, ω úhlová rychlost, B koeficient viskosního tření, J celkový moment setrvačnosti na hřídeli a M_0 moment odporu. Rovnice (2.1) představuje rovnováho napětí v obvodu kotvy a rovnice (2.2) je pak rovnováho momentů na hřídeli motoru.

Po dosazení $C\Phi\omega = u_i$ má diferenciální rovnice (2.1) po Laplaceově transformaci tvar

$$U = R_a I(p) + pL_a I(p) + U_i(p) \tag{2.3}$$

Přenos $F_1(p)$ je poměr výstupu ke vstupu: výstupem je zde proud *I(p)*, vstupem potom rozdíl přiváděného napětí na kotvu a napětí indukovaného U-U_i . První část modelu motoru má tvar

$$F_1(p) = \frac{I(p)}{U - U_i(p)} = \frac{1}{R_a + pL_a} = \frac{\frac{1}{R_a}}{1 + p\tau_a}, \qquad \tau_a = \frac{L_a}{R_a} \tag{2.4}$$

Dalším členem modelu je proporcionální člen, vyjadřující závislost momentu motoru na proudu kotvy $m = C\Phi i$; v operátorovém tvaru $M(p) = C\Phi I(p)$. Přenos tohoto členu je

$$F_2 = \frac{M(p)}{I(p)} = C\Phi \tag{2.5}$$

Rovnice rovnováhy momentů (2.2), po zanedbání viskosního tření B, je v operátorovém tvaru $M(p) = pJ\omega(p) + M_0$.

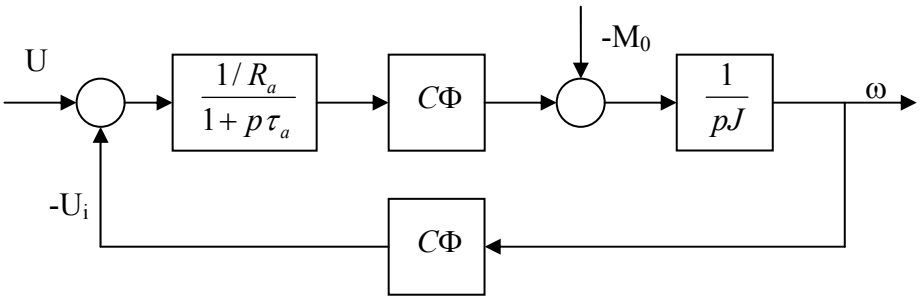
Přenos členu s momentem setrvačnosti J má integrační charakter

$$F_3(p) = \frac{\omega(p)}{M(p) - M_0} = \frac{1}{pJ} \tag{2.6}$$

Závislost mezi indukovaným napětím v kotvě a úhlovou rychlostí je lineární, přenos tohoto členu je proporcionální

$$F_4(p) = \frac{U_i(p)}{\omega(p)} = C\Phi \tag{2.7}$$

Přenosové funkce $F_1(p)$, $F_2(p)$, $F_3(p)$ a $F_4(p)$ tvoří matematický model stejnosměrného motoru s konstantním magnetickým tokem, jehož blokové schéma je na *Obr.2.6*.



Obr. 2.6 Matematický model stejnosměrného motoru

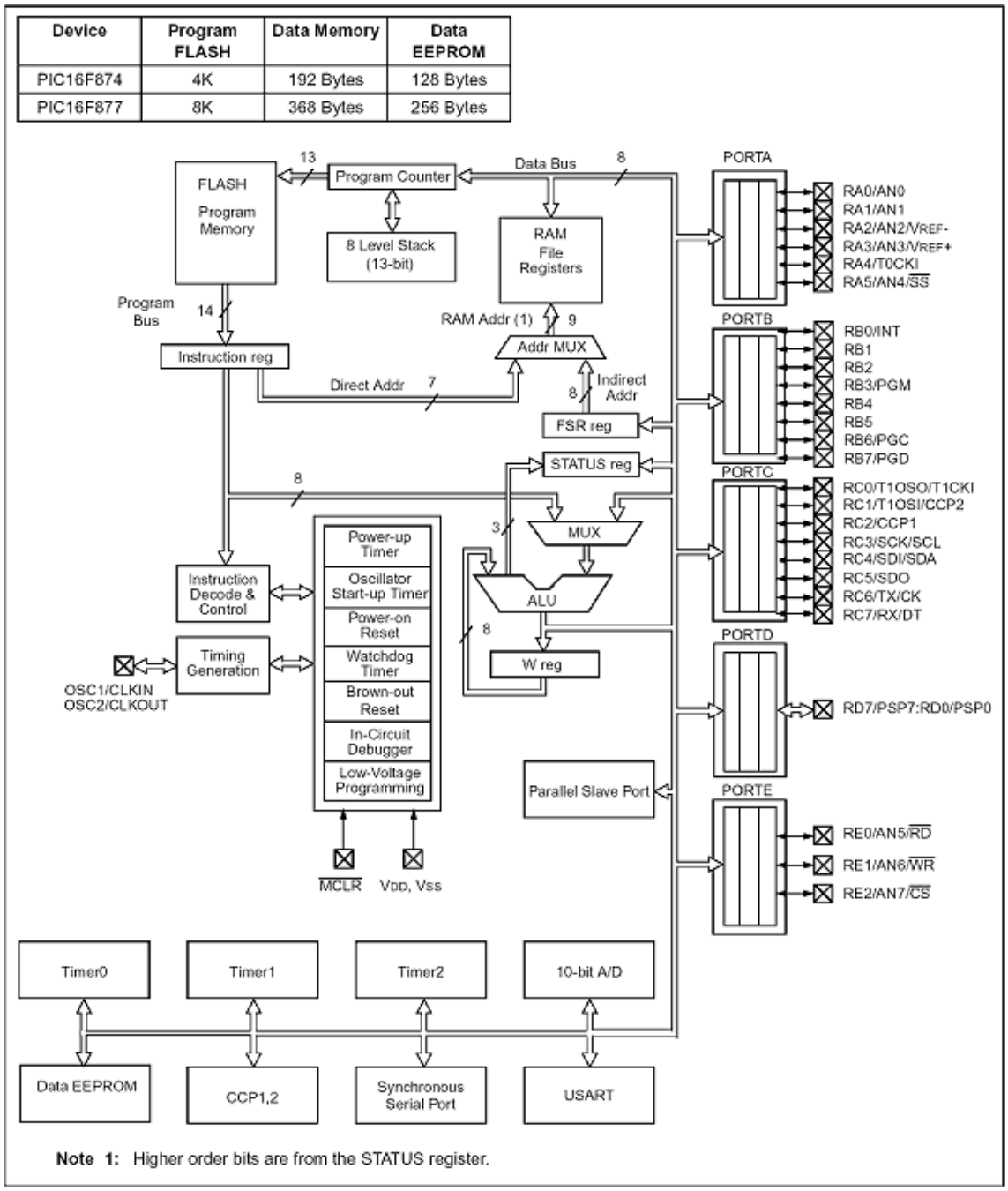
[1, 2, 6]

3. Jednočipové mikropočítače (mikrokontroléry)

Jednočipové mikropočítače jsou stále častěji základem většiny řídících a měřících obvodů a mají obrovské množství aplikací v přístrojích, které jsou vybaveny schopností automaticky pracovat podle předem daného programu. Aplikační rozsah těchto řídicích systémů je prakticky neomezený. Má využití od běžných elektronických přístrojů používaných v domácnosti až po nejnáročnější aplikace v různých odvětvích průmyslu a ve vojenství. Vývoj a aplikace jednočipového mikropočítače je realizován v řadě modifikací, vhodných pro dané specifické aplikace. Mezi nejvíce rozšířené mikrokontroléry z hlediska aplikací patří mikroprocesory výrobců, a to Motorola, Microchip, Atmel a Intel. Při vývoji nových jednočipových mikropočítačů není hlavním cílem zvýšení rychlosti a kapacity jako je tomu například u osobních počítačů, ale především vhodnost na danou aplikaci. Toto vše vede k tomu, že je na trhu široká nabídka mikrokontrolérů, ze které si lze vybrat ten nejvhodnější a docílit tak co nejnižších nákladů a zjednodušení řízených aplikací. Jak již bylo zmíněno, existuje široká škála jednotlivých typových představitelů, vzájemně se lišících implementovanými technickými prostředky, velikostí EPROM (EEPROM, FLASH) a RAM, počtem a typem I/O pinů, frekvenčním rozsahem, typem oscilátoru, pouzdry, teplotním rozsahem apod. Je také velká snaha o integrování různých specializovaných obvodů pro řízení periferií, které pak minimálně zatěžují centrální jednotku mikrokontroléru. Mezi takové specializované obvody patří například zrychlení obsluhy přerušení, tzv. jednotka zpracování událostí a jiné.

Jednotlivé generace jednočipových mikropočítačů se tak liší počtem časovačů, počtem přerušovacích vstupů, přítomností A/D převodníku na čipu a jeho rozlišením, případně přítomností speciálních generátorů řídicích signálů jako je pulzní šířková modulace nebo možností přímého připojení k rozhraní RS232 nebo I^2C.

[8, 9, 10]



Obr.3.1 Architektura mikrokontroléru PIC16F877

3.1 Vnitřní uspořádání mikrokontroléru

Mikrokontroléry, jinak též označované jako jednočipové mikropočítače, obsahují v jediném pouzdře všechny podstatné části mikropočítače:

- Řadič a aritmetickou jednotku (ALU). Podle typu mikrokontroléru se používá délka slova 4, 8, 16 nebo 32 bitů.

- Paměť programu (kódovou paměť). Paměť programu je buď typu EPROM nebo Flash, u mikrokontrolérů vyráběných pro určitou konkrétní aplikaci s pevně daným programem pak typu ROM.
- Paměť typu Read/Write, někdy doplněná i o paměť EEPROM. Paměť typu EEPROM má velkou výhodu v tom, že si uchovává svůj obsah i po odpojení napájení.
- Periferní obvody pro vstup s výstup dat.

Dále mikrokontroléry obsahují generátor hodinového signálu a další technické prostředky, jako jsou obvody pro kontrolu správné činnosti mikrokontroléru, obvody programování kódové paměti přímo v aplikaci, A/D a D/A převodníky, řadiče přerušení, DMA řadiče apod.

Na *Obr.3.1* je blokové schéma mikrokontroléru PIC16F877, který je použit při realizaci řízení laboratorního modelu portálového jeřábu.

[7, 8, 9, 10]

3.2 Organizace paměti

Co se týče organizace paměťových adresních prostorů, lze rozdělit mikrokotroléry do dvou skupin:

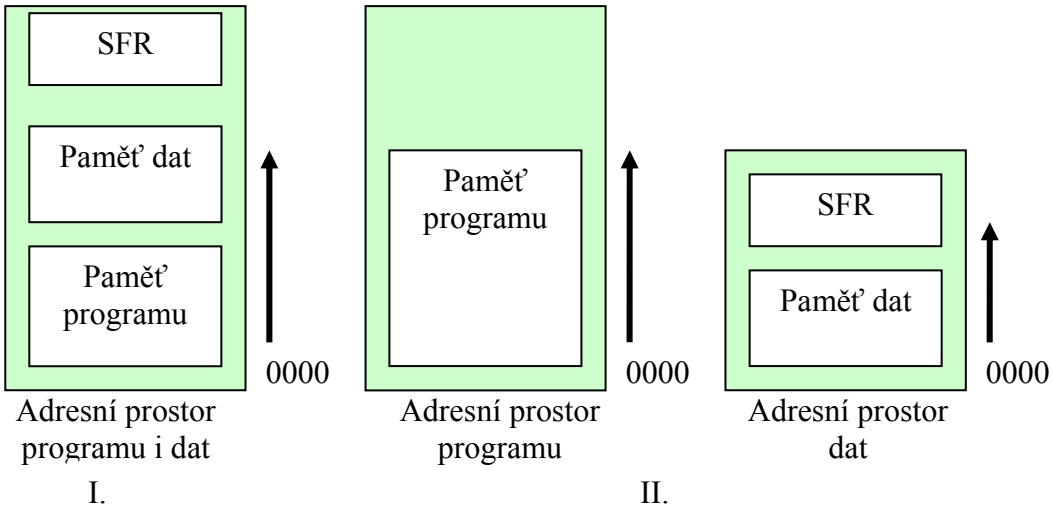
I. Mikrokontroléry s lineárním uspořádáním adresního prostoru (*Obr.3.2I*). Zde je použit jediný adresní prostor pro mapování paměti programu, paměti dat i registrů pro řízení I/O obvodů.

   Tuto architekturu používají např. mikrokontroléry Motorola nebo Intel 80196.

II. Mikrokontroléry s odděleným adresním prostorem pro paměť programu a paměť dat, označována jako Harvardská architektura *(Obr.3.2II).* Harvardská architektura je charakteristická pro mikrokontroléry firmy Microchip, Atmel nebo Intel 8051,8052 a mnoho dalších.

Kapacita paměti se u různých typů mikrokotrolérů pohybuje v rozmezí od 1kB do 256kB pro paměť programu a od 32 bytů do 16kB pro paměť dat.

[8, 9]



Obr.3.2 Lineární uspořádání (I.) a Harvardská architektura (II.) , SFR – □peciál Function Registers (registry pro řízení I/O obvodů).

3.3 Periferní obvody

Vybavení mikrokontrolérů periferními obvody je velmi různorodé. Často se vyrábí určitý typ mikrokontroléru v mnoha variantách, které se liší právě periferními obvody. Svými charakteristikami jsou jednotlivé varianty určené pro jistou konkrétní aplikaci, například řízení elektrických pohonů, ovládání komunikačních zařízení apod.
Za standardní lze považovat následující periferní obvody mikrokontroléru.

Základní jsou **Paralelní I/O** porty, které slouží pro vstup a výstup binárních dat. Obvykle lze pracovat s celým portem najednou nebo samostatně ovládat jednotlivé piny.

Mezi další standardní obvody patří **Obvody čítačů a časovačů**, které jsou v základní konfiguraci s jedním až dvěma 8 nebo 16 bitovými čítači/časovači, ty mohou být u specializovaných mikrokontrolérů rozšířeny až na šest čítačů. Čítače mohou být potom vybaveny speciálními funkcemi, které umožňují zachytávání změny logické úrovně signálů na vybraných vstupech, generování obdélníkových průběhů signálu, dekódování signálu polohových čidel a jiné.

Pro komunikaci s okolím slouží mikrokontrolérům **Sériové rozhraní**. Jednotlivé mikrokontroléry mohou být vybaveny až pěti sériovými komunikačními linkami. Kromě standardního asynchronního přenosu dat (RS232) často tyto linky podporují i jiné protokoly jako jsou například I^2C, SPI, CAN a podobně. Sériová rozhraní s těmito protokoly jsou používána pro připojování vnějších pamětí se sériovým

rozhraním, vnějších A/D převodníků, obvodů pro měření času (Real Time Clock) atd.

Většina vyráběných mikrokonrolérů nemá dostatek pinů potřebných k připojení všech vstupních a výstupních signálů, proto má velké množství pinů více funkcí. V daném programu se pak může použít pouze jedna z nich. Vícenásobné použití pinů mikrokontroléru je velmi dobře vidět na *Obr.3.1*. Některé vývody portů mikrokontroléru PIC16F877 mají tři až čtyři různé funkce.

[8, 9]

3.4 Mikrokontroléry Microchip řady PIC16Fxx

Jsou to univerzální 8-mi bitové jednočipové mikrokontrolery. Všechny tyto řadiče jsou vyrobeny technologií CMOS (complementary metal-oxide semiconductor) a jsou založeny na rozšířené architektuře RISC (Reduced Instruction Set Computer). Mají oddělenou programovou a datovou paměť (Harvardská architektura). Vnitřní systém redukuje nutnost připojení externích obvodů na minimum, čímž zlevňuje konečné aplikace.

Základní charakteristiky mikrokonroléru PIC16Fxx jsou následující :

- Používají sadu 35 instrukcí, kde instrukce používají jeden strojový cyklus, při odskoku a návratu dva cykly.
- Provozní frekvence je až 20MHz (200ns = jeden instrukční cyklus)
- 14-bitové instrukce (paměť programu flash)
- 8-bitová data (paměť dat RAM a EEPROM(stálé i bez napájení))
- 8-úrovňový hardwarový zásobník
- Rozsah pracovního napětí 2,0V – 6,0V.
- Velké zatížení vývodů 25mA.
- Přímý, nepřímý a poměrný adresový režim.

V každém mikrokontroléru jsou k dispozici speciální funkční registry, které slouží k nastavení určitých vlastností mikrokontroléru nebo odečítání speciálních stavů. U jednočipových mikropočítačů firmy Microchip to jsou především registry:

- STATUS : Tento registr obsahuje aritmetický stav ALU, stav RESETu a bity vybírající banku pro paměť dat.
- OPTION_REG : Do tohoto registru lze zapisovat i z něho lze číst. Obsahuje různé kontrolní bity pro konfiguraci předděliče TMR0 / WDT, externího INT přerušení, TMR0 a přerušení na PORTuB.

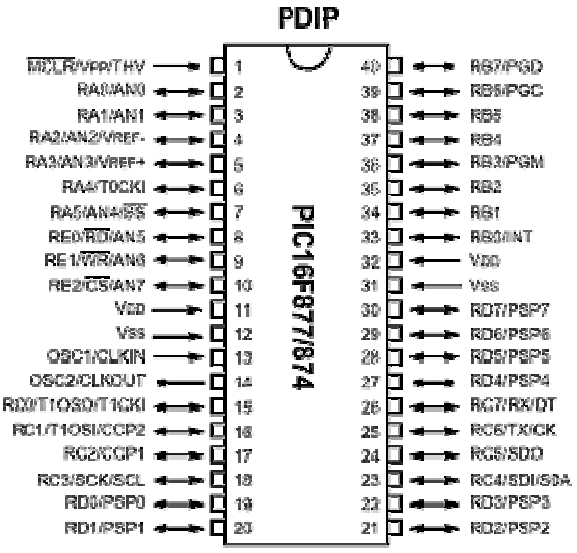
- INTCON : Z tohoto registru lze také číst a zároveň do něho zapisovat. Obsahuje různé varianty zapnutí a detekcí stavů pro přetečení TMR0 registru, změn na PORTuB a externího INT přerušení.
- ADCON0: Kontroluje operace s A/D převodníkem.
- ADCON1: Konfiguruje funkci pinů, mohou být nastaveny jako analogové vstupy nebo digitální I/O .
- PIE1: Tento registr obsahuje individuální aktivační bity pro vnější přerušení.
- PIE2 : Tento registr obsahuje aktivační bity pro CCP2 vnější přerušení, kolizi na sběrnici SSP a přerušení při zápisu do EEPROM.
- PIR1: Z tohoto registru se čtou aktuální stavy jednotlivých externích přerušení.
- PIR2: Viz. PIE2, ale místo aktivace jsou v něm uloženy aktuální stavy.
- EECON1 : Je kontrolní registr pro konfiguraci a inicializaci přístupu k EEPROM paměti.
- EECON2 : Toto není fyzicky implementovaný registr, ale je používán výlučně při sekvenčním paměťovém zápisu.
- EEADR : Obsahuje cílovou adresu v EEPROM.
- EEDATA: Obsahuje data, která mají být na cílovou adresu zapsána.
- EEADRH a EEDATAH : Slouží k rozšíření adresovacího a datového prostoru.

Při realizaci laboratorního modelu portálového jeřábu jsou použity dva mikrokontroléry PIC16F84A/20 a jeden PIC16F877/20. Při dalším krátkém popisu se zaměřím na druhý z nich , který má mnohem větší výbavu a tím je i více univerzálnější.

[4, 5, 7, 8, 10]

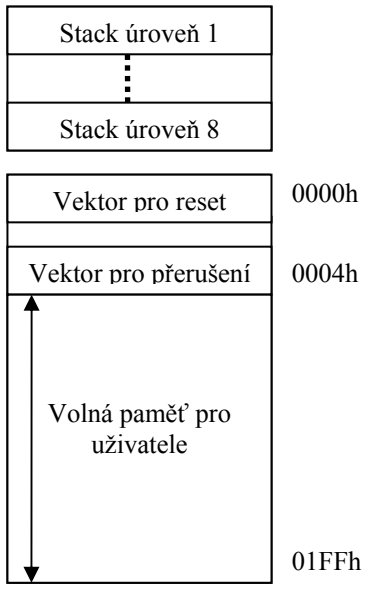
3.4.1 Mikrokontrolér PIC16F877

Architektura je patrná z *Obr.3.1*, kde aritmetická jednotka (ALU) dovoluje sčítat, odčítat, posouvat obsah registru a provádět logické operace. Aritmetické operace mají dva operandy, kde jeden z nich je uložen v pracovním registru (W-registr) a druhý je buď konstanta nebo registr v paměti. Při vykonávání jednotlivých instrukcí ALU jsou ovlivňovány hodnoty příznaků v registru STATUS.



Obr.3.3 Rozložení pinů u PIC16F877

Tento mikrokotrolér je vybaven třemi časovači / čítači (dvěma 8-bitovými a jedním 16-bitovým, který může být použit jako zdroj pro vnější oscilátor), dvěma záchytnými, porovnávacími PWM (pulsní šířková modulace) moduly s maximálním rozlišením 10 bitů, 10-bitovým A/D převodníkem, synchronním sériovým portem SSP s SPI (master mód) a I^2C (master / slave), univerzálním synchronním asynchronním přijímačem / vysílačem (USART / SCI) s 9-bitovou detekcí a paralelním 8-bitovým slave portem (PSP). Rozložení a popis jednotlivých vývodů pouzdra (pinů) je patrné z *Obr.3.3*.



Obr.3.4 Rozložení paměti programu pro PIC16F877

Paměť programu procesoru PIC16F877 (*Obr.3.4*) má 13-bitový programový čítač (PC), který adresuje paměť 8Kx14bitů (0000h-01FFh). Po resetu začíná procesor na adrese 0000h a po přerušení na adrese 0004h.

Paměť dat je rozdělena do čtyř bank (prostorů) o velikosti 128B statické RAM, které obsahují základní funkční registry (00h-1Fh) a registry volné pro uživatele (20h-7Fh). Některé z těchto registrů jsou adresovány ve všech bankách.

Mikrokontrolér PIC16F877 má 5 portů PORTA, PORTB, PORTC, PORTD a PORTE, které mají dohromady 33 vývodů. Porty se dají nastavit v registrech TRISA - TRISE jako vstupní nebo výstupní podle požadavku programu.

Všechny I/O vývody mají řídící bit pro směr toku dat (v TRIS registru), kterým může být každý z vývodů nastaven jako vstup nebo výstup nezávisle na ostatních.

Nastavením příslušného bitu na "1" v registru TRIS se nastaví příslušný vývod jako vstupní, nastavením příslušného bitu na "0" se nastaví příslušný vývod jako výstupní.

TTL logika definuje, že napětí 5V reprezentuje logickou jedničku a napětí 0V reprezentuje logickou nulu. Vývody typu ST (Shmittův klopný obvod) využívají dvou prahových hodnot pro detekci logické jedničky a nuly. Pokud je na pinu logická jednička je třeba s napětím klesnout např. pod úroveň 2V pro přepnutí do logické nuly. Obráceně, pro přepnutí z logické nuly je třeba napětí zvýšit až třeba na 3V. Drobné zákmity kolem jednotlivých hranic tedy nezpůsobují falešné přechody (zákmity).

PORTA je 6-bitový. Vývod RA4 má na vstupu Schmittův klopný obvod a jako výstup má otevřený kolektor. Všechny ostatní vývody PORTuA mají jako vstupní úrovně TTL a jako výstup budiče CMOS. Vývod RA4 je možné přepnout jako vstup pro hodinový signál TMR0. Všechny ostatní vývody PORTuA jsou připojeny na multiplexor A/D převodníku. Nastavení převodníku se provádí v registru ADCON1.

PORTB je 8-bitový. Každý z vývodů portu B může mít programově připojen slabý vnitřní pull-up odpor (cca 100 μA) na všech vývodech konfigurovaných jako vstupní. Což znamená, že při odpojeném vývodu je na něm stálé napětí 5V. Naopak, pokud je vývod připojen na zem, je na něm napětí 0V. Toto je automaticky vypnuto u těch vývodů, které jsou nastaveny jako výstupní. Čtyři vývody PORTuB (RB4 - RB7), pokud jsou nastaveny jako vstupní, můžou vyvolat přerušení při změně stavu.

PORTC je 8-bitový. Na vývodech RC3 a RC4 je vyvedena sběrnice I^2C a na vývodech RC6 a RC7 je vyveden univerzální synchronní asynchronní přijímač / vysílač. Celý port má na vstupech Schmittův klopný obvod.

Port D je 8-bitový. Celý port má stejně jako PORTC na vstupech Schmittův klopný obvod. PORTD může být nakonfigurován jako 8-bitový paralelní slave port (PSP), při tomto nastavení jsou pak jednotlivé piny v TTL logice. Poslední port PORTE je 3-bitový.

Bližší informace k mikrokontroléru PIC16F877 lze nalézt v katalogovém listu, který je k dispozici na webových stránkách výrobce tohoto čipu nebo na přiloženém CD.

[4, 5, 7]

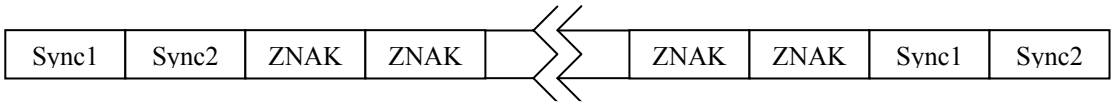
3.5 Komunikace mikrokotroléru s okolím

Komunikace systému s jinými systémy (jinými mikrokontroléry, počítači, inteligentními obvody apod.) bývá nejčastěji realizována pomocí nějakého standartu. Pro omezení použitých I/O vývodů (pinů) je nejvíce využívána sériová komunikace. Sériovou komunikaci rozdělujeme na synchronní a asynchronní.

3.5.1 Synchronní komunikace

Principiálně nejjednodušší synchronní přenos informací znamená, že na nějakém vodiči nebo vodičích se nastaví určitá úroveň, která přenáší informaci a validita informace se potvrdí impulzem, nebo změnou úrovně synchronizačního signálu. Synchronizačním signálem se tedy informace kvantují. Toto zapojení ovšem vyžaduje samostatný přenosový kanál (vedle kanálu datového), který nebývá často k dispozici.

Druhým řešením je smíchat hodinový signál se signálem, který reprezentuje samotná vysílaná data, a přenášet vše jediným společným přenosovým kanálem. Synchronizace přijímače a vysílače na začátku přenosu tedy zajišťuje synchronizační znak. Po jeho vyhodnocení probíhá nepřerušovaný přenos znaků bez další synchronizace až do okamžiku přijetí nového synchronizačního znaku. Pokud vysílač nemá k dispozici další slovo dat, vysílá automaticky synchronizační znaky.



Obr.3.5 Synchronní přenos dat

Tento typ přenosu je méně častý, protože je velice přísně vázaný na přesnost generování délky znaků.

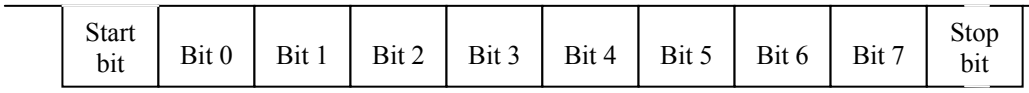
[4, 5, 10]

3.5.2 Asynchronní komunikace

Asynchronní přenos dat přenáší data v určitých sekvencích. Data jsou přenášena přesně danou rychlostí a uvozena startovací sekvencí, na kterou se synchronizují všechna přijímací zařízení. Všechny strany obsahují vlastní přesný oscilátor, díky kterému odečítají data v přesně definovaných intervalech. Po ukončení sekvence je další příjem opět synchronizován startovací sekvencí.

Základní tvar při asynchronním přenosu dat začíná start bitem. Po něm následuje osm datových bitů, vysílaných od bitu 0 po bit 7. Jako poslední je vysílán stop bit, který může mít pohyblivou šířku, kterou je možné definovat v řídícím slově sériového

přenosu. Stop bit může potom mít délku 1; 1,5 nebo 2 bity. Delší stop bit se používá u pomalejších zařízení pro doběh zpracování přijatého znaku. Přenos dalšího slova je opět zahájen start bitem.



Obr.3.6 Sériový asynchronní přenos znaku

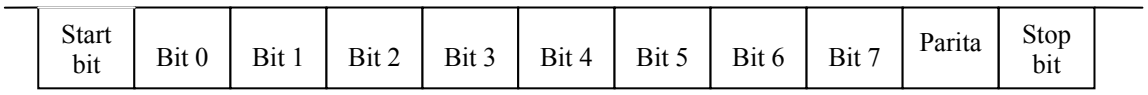
Délka start bitu a datového bitu se určí z požadované přenosové rychlosti následovně:

doba trvání bitu (s) = 1 / přenosová rychlost (Bd)

Přenosová rychlost se udává v Bd „baudech“ (počet bitů za sekundu). Standardně jsou používány rychlosti 2,4; 4,8; 9,6; 19,2; 38,4 kBd. Přenosová rychlost se volí podle maximální délky vedení, kde například pro rychlost 19 200 Bd je doporučována maximální délka 15 metrů.

Pokud je nutné zabezpečit správnost přenosu dat, používá se k tomu tzv. parita. Ve vysílacím zařízení se sečte počet jedničkových bitů a doplní se paritním bitem tak, aby byla zachována předem dohodnutá podmínka sudého nebo lichého počtu jedničkových bitů.

- Sudá parita – počet jedničkových bitů + paritní bit = sudé číslo
- Lichá parita – počet jedničkových bitů + paritní bit = liché číslo



Obr.3.7 Sériový asynchronní přenos znaku s paritou

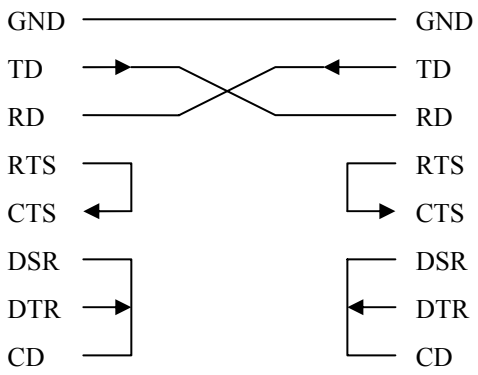
[4, 5, 10]

3.5.3 Komunikace s rozhraním RS232

RS232 je rozhraní pro přenos informací vytvořené původně pro komunikaci do vzdálenosti 20 m. Pro větší odolnost proti rušení je informace po propojovacích vodičích přenášena větším napětím, než je standardních 5 V. Přenos informací probíhá asynchronně, pomocí pevně nastavené přenosové rychlosti a synchronizace sestupnou hranou startovacího impulzu.

Při komunikaci mikrokontroléru s okolím probíhá přenos jedním směrem po jednom vodiči (nepočítá-li se společný vodič GND). Vysílací strana je označována TD (TxD – transit data) a přijímací strana RD (RxD – recieve data). Každé zařízení, které

má tedy komunikovat obousměrně, má samostatný vysílač a přijímač. Klasické rozhraní RS232 je určeno pro připojení modemu a může používat ještě jiné signály než jen TD a RD (viz. *Tabulka 3.1*). Do vzdálenosti 15 m však lze používat připojení jako „nulový modem“ a tedy pouze dva datové vodiče plus signálovou zem GND. Takovéto zapojení používají i mikrokontroléry firmy Microchip (například PIC16F877).



Obr.3.8 3-drátové propojení

Stavy logické nuly a logické jedničky pro rozhraní RS232 jsou definovány napěťovými úrovněmi. Pro větší odolnost proti rušení není použito standardních hodnot do 5V. Logická 1 (marking state) je vysílána v rozsahu –5 až –15V a přijímána v rozsahu –3 až –25V. Rozsah na straně přijímače je širší z důvodu možného rušení signálu během přenosu. Logická 0 (space state) je kódována stejnými úrovněmi s opačnou polaritou. Pásmo –3V až +3V je považováno za nedefinovaný stav. Výstupem z procesoru PIC je ovšem signál na logických úrovních TTL, tedy do 5V. Aby bylo možné komunikovat s rozhraním RS232, je nutné tyto úrovně konvertovat na výše zmíněné hodnoty.

PIN	NÁZEV	SMĚR	POPIS
1	CD	←	Modem oznamuje terminálu, že na telefonní lince detekoval nosný kmitočet.
2	RD	←	Tok dat z modemu do terminálu .
3	TD	→	Tok dat z terminálu do modemu .
4	DTR	→	Terminál tímto signálem oznamuje modemu, že je připraven komunikovat .
5	GND	←→	Signálová zem
6	DSR	←	Modem tímto signálem oznamuje terminálu, že je připraven komunikovat.
7	RTS	→	Terminál tímto signálem oznamuje modemu, že komunikační cesta je volná.
8	CTS	←	Modem tímto signálem oznamuje terminálu, že komunikační cesta je volná .
9	RI	←	Indikátor zvonění. Modem oznamuje terminálu, že na telefonní lince detekoval signál zvonění.

Tabulka 3.1. Popis signálů rozhraní RS232 pro konektor CANNON 9

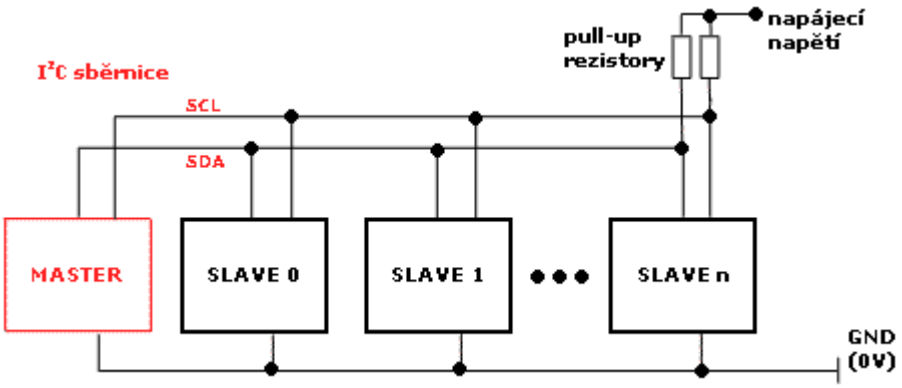
3.5.4 Komunikace pomocí sběrnice I^2C

Sběrnice I^2C (Inter – IC bus) je dvouvodičové datové propojení mezi jedním nebo několika procesory (master) a speciálními periferními součástkami (slave). Všechny součástky jsou připojeny na tutéž sběrnici a je možno se na ně cíleně obracet prostřednictvím jejich adres. Adresy a data se přenášejí týmiž linkami. Tato sběrnice umožňuje jednoduché propojení mnoha integrovaných obvodů a bezproblémové dodatečné rozšiřování. Připojit je možno všechny integrované obvody, které ovládají speciální protokol sběrnice. Kromě pamětí RAM, EEPROM, součástek pro rozšíření portů, A/D a D/A převodníků a obvodů hodin se dodává řada speciálních obvodů, například budiče displejů nebo integrované obvody pro televizní techniku.

Přenosová rychlost sběrnice je pro většinu aplikací dostatečná i v základní verzi, kde je frekvence hodin 100kHz. Ve vylepšených verzích to může být 400kHz nebo 1MHz, ale ne všechny integrované obvody tuto verzi podporují. Rychlost přenosu pak musí být přizpůsobena pochopitelně "nejpomalejšímu" čipu na sběrnici. Oba vodiče musí být implicitně v logické jedničce, a to je zajištěno pull-up rezistory. Jejich odpory mají hodnotu v řádech jednotek kiloohmů. Čím je vyšší komunikační frekvence, tím musí být nižší hodnoty těchto odporů.

Princip přenosu I^2C

Jeden z integrovaných obvodů (většinou mikrokontrolér) je nastaven jako MASTER a všechny ostatní obvody jsou SLAVE. Obvody se dají zapojit i jako tzv. multi-master, kdy je čipů master několik. Sběrnice používá sériovou datovou linku SDA a hodinovou (taktovací) linku SCL. Data a adresy se podobně jako u posuvných registrů přenášejí společně s hodinovým taktem. Obě tyto sběrnice je možno používat pro přenos dat v obou směrech. Master při jakémkoli přenosu generuje hodinový signál na vodiči SCL. Když jeden čip vysílá, přijímají všechny ostatní a pouze podle adresy určují, zda jsou data určena jim. Čip, který chce vyslat/přijmout data musí nejprve definovat adresu čipu, s kterým chce komunikovat a zda půjde o příjem nebo vysílání - tedy o čtení nebo zápis. To určuje R/W (read/write) bit, který je součástí adresy.

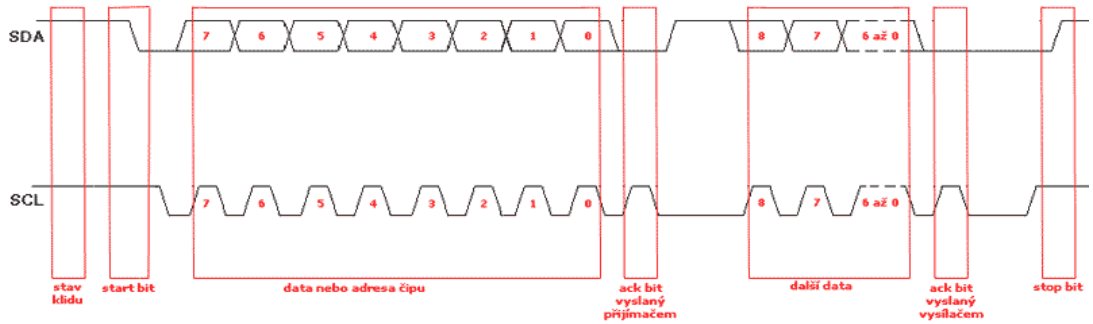


Obr.3.9 Princip zapojení jednotlivých integrovaných obvodů pomocí sběrnice I^2C

Protokol sběrnice I^2C má řadu přesně definovaných situací, které každému účastníkovi sběrnice umožňují rozpoznat začátek a konec přenosu a také adresy.

- **stav klidu** - Je zajištěn logickými jedničkami na obou vodičích, master tedy negeneruje hodinový signál a neprobíhá žádný přenos. Logické jedničky jsou na obou vodičích zajištěny pull-up rezistory (rezistory mezi vodičem a napájecím napětím), takže klidový stav nastane, i pokud jsou výstupy obvodu master ve stavu vysoké impedance (tedy odpojeny).
- **start bit** - Zahajuje přenos nebo jeho další část. Je vygenerován tak, že se změní úroveň SDA z 1 na 0 zatímco je SCL v logické 1.
- **stop bit** - Ukončuje přenos. Je vygenerován podobně jako start bit. Logická úroveň SDA se změní z 0 na 1 zatímco je SCL v logické 1. Stop bit může být generován pouze po "nepotvrzení přenosu", tedy pouze po přijmutí Ack v logické 1. (viz níže)
- **přenos dat** - Data jsou přenášena po 1Byte tedy 8 po sobě jdoucích bitů od nejvyššího po nejnižší. Při přenosu dat se může logická úroveň na SDA měnit pouze pokud je SCL v logické 0. Při každém pulzu na SCL je přenesen jeden bit.
- **potvrzující bit Ack** (acknowledge) - Tento bit slouží k potvrzení správného přijmutí dat. Ack bit se odesílá stejným způsobem jako by se odesílal devátý bit dat, ale s tím rozdílem, že ho generuje čip, který přijímal (přijímač) a nikoliv ten, který data odesílal. Pokud přenos proběhl v pořádku, tak odešle logickou 0. Logická 0 potvrzujícího bitu znamená rovněž to, že je přijímač připraven na příjem dalšího byte, který následuje okamžitě po něm při dalším pulzu na SCL. Pokud přenos selhal, odešle logickou 1. Nebo pokud má dojít k ukončení

přenosu, tak "neodešle nic". Pull-up rezistor pak zajistí, že bude na SDA logická 1 a Ack bit (v logické 0) odešle vysílač.



Obr.3.10 Časový průběh logických úrovní na vodičích SDA a SCL

[4, 5, 10]

3.6 Programování mikrokontrolérů

Program se skládá z instrukcí, kde každá instrukce má svůj kód. Jedná se o binárně kódovaný příkaz, který se zapisuje pomocí hexadecimálních čísel. Tento zápis instrukcí se nazývá strojový kód.

Řada binárních nebo hexadecimálních čísel je pro lidské chápání velmi nepřehledná, proto se tato komunikace usnadňuje pomocí mnemotechnických zkratek, které pocházejí z angličtiny a které nahrazují instrukce strojového kódu (Např. MOV = přesun dat, SUB = odečítání atd.). Operandům (datům) se přiřazují názvy stejně jako důležitým adresám v paměti. Takováto substituce se nazývá Jazyk Symbolických Adres (JSA). To vše umožňuje programování na té nejnižší úrovni, tzn. co nejblíže mikroprocesoru, a tím i maximální využití jeho rychlosti i kapacity.

Program napsaný v Jazyku Symbolických Adres je přeložen do strojového kódu pomocí překladače, který se nazývá ASSEMBLER, který se může u jednotlivých výrobců mikrokontrolérů lišit. Stejně tak se může lišit počet instrukcí, které určitý mikrokontrolér umí. V praxi se někdy používá název ASSEMBLER i pro samotný JSA.

Zápis jazyka ASSEMBLER se skládá ze dvou částí, a to z vlastního kódového slova instrukce a tzv. operandu. Například pro zápis dat registru W do registru se symbolickou adresou *MojeAdresa* se napíše:

MOVWF *MojeAdresa,f*

Z uvedeného zápisu je zřejmé, že výraz MOVWF jé kódové slovo instrukce a výraz *MojeAdresa,f* je operand. Kódové slovo instrukce jednoznačně určuje, co se bude

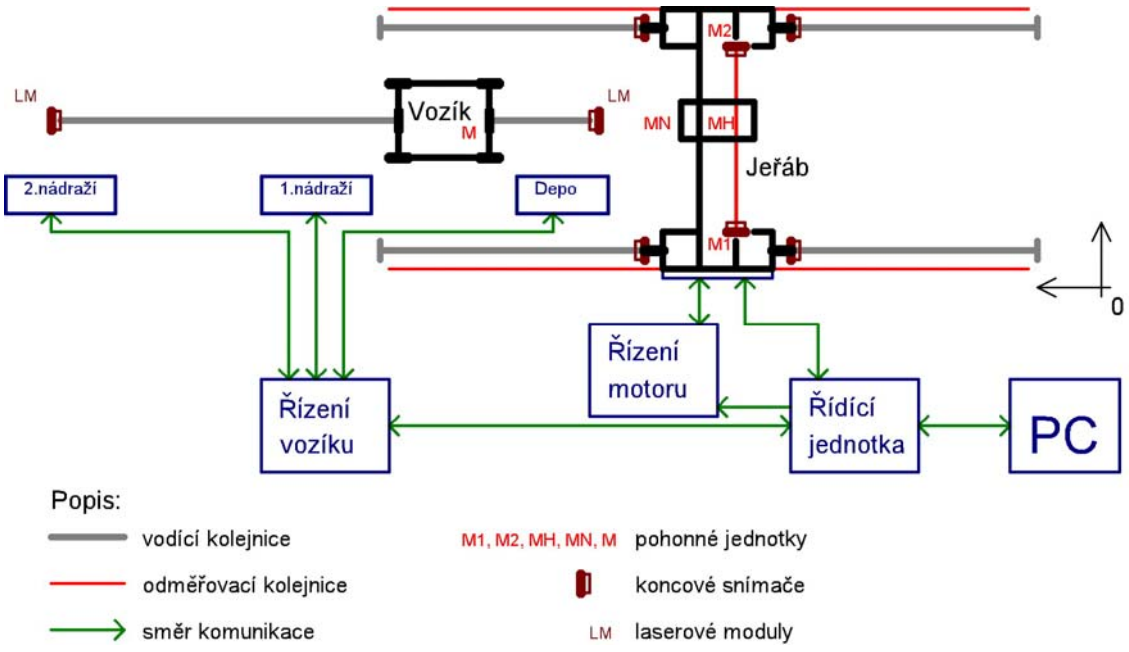
provádět za operaci a operand určí, kde a s jakými registry či hodnotami se bude operace provádět, případně kam se výsledek této operace uloží. Z toho plyne, že se může formát operandu u jednotlivých instrukcí lišit. Seznam s popisem všech instrukcí pro mikrokontroléry firmy Microchip se nachází na přiloženém CD.

Po fyzické stránce se k programování používají různé programátory nebo se některé mikrokontroléry, pokud to umožňují, mohou programovat přímo přes tzv. ICSP programování, které umožňuje nechat mikrokontrolér v zapojené desce. Díky tomu nehrozí poškození pouzdra mikropočítače, které může vzniknout při opakovaném vyndání mikrokontroléru z patice.

[4, 5]

4. Realizace – systém pro přepravu materiálu (výrobků)

Celý systém má za úkol přepravu materiálu (nebo výrobků) z jednoho místa na druhé. Nadřazeným řídícím systémem je v tomto případě osobní počítač, na kterém běží program, kterým je celý polohový mechanismus ovládán. Po zadání požadované polohy, které má být dosaženo, předá PC pomocí sériového rozhraní data do řídícího mikropočítače portálového jeřábu, který dál komunikuje s řídícím mikropočítačem vozíku a navede ho do žádané polohy. Po dosažení určené polohy potvrdí řídící mikropočítač jeřábu doručení materiálu (popř. výrobků) na místo určení. Osobní počítač je v tomto systému brán jako nadřazený řídící systém a kromě určování požadované polohy má také za úkol graficky znázornit aktuální stav celého systému. Uspořádání a komunikace celého systému je patrné z *Obr.4.1*.

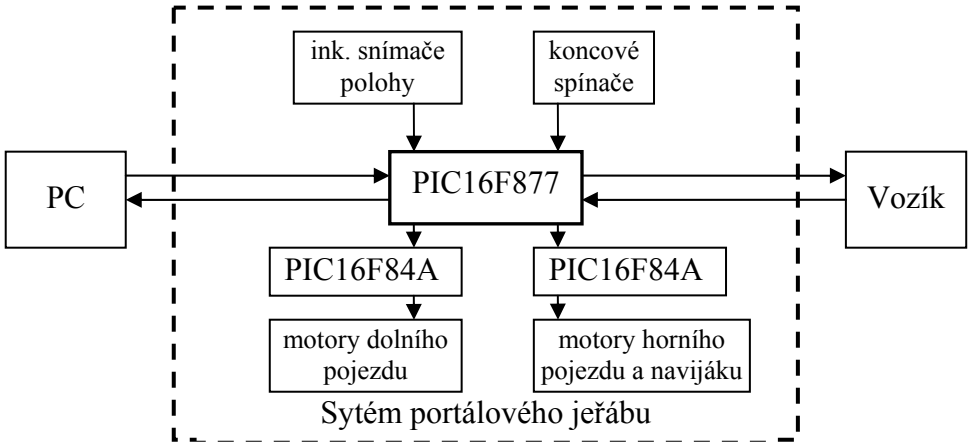


Obr.4.1 Schéma uspořádání sytému pro přepravu materiálu

4.1 Uspořádání modelu portálového jeřábu

Laboratorní model portálového jeřábu se skládá z řídícího mikropočítače (mikrokontrolér PIC16F877), pomocí kterého se realizuje regulace a komunikace s okolím, dvou mikropočítačů (PIC16F844A), které ovládají pomocí pulzní šířkové modulace motory pro jednotlivé pohyby jeřábu, inkrementálních snímačů polohy,

koncových spínačů, čtyř stejnosměrných motorků a samotné mechaniky sestavené pomocí stavebnice Merkur. Celé schéma je patrno z *Obr.4.2.*

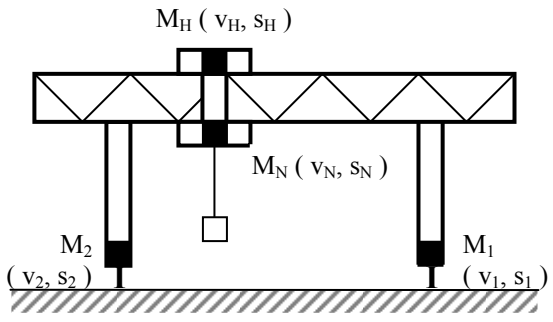


Obr. 4.2 Schéma uspořádání a komunikace modelu portálového jeřábu

4.2 Princip chování modelu

Celý model jeřábu se pohybuje po kolejích, podél kterých vedou ještě pomocné kolejnice, které slouží k inkrementálnímu odměřování polohy. Po jeřábovém mostu se pohybuje menší pojezd, jehož pohyb je kolmý k hlavním kolejím. Tento pojezd je poháněn motorem M_H a je na něm zavěšen navíjecí mechanismus, který je realizován pomocí motoru M_N. Převod je zde uskutečněn pomocí ozubených kol přímo na navíjecí tyč. Na konci závěsného lana je připevněn elektromagnet, pomocí kterého lze snadno uchopit malé kovové předměty.

V praxi při pohybu portálového jeřábu s rozpětím několika desítek metrů vznikají těžkosti se vzpříčením jeřábového mostu, které je důsledkem hlavně nerovných tlaků na jednotlivé nohy jeřábu. Podélný pohyb jeřábu proměnlivou rychlostí po kolejnicích je zabezpečovaný při náročnějších zařízeních stejnosměrnými motory, resp. skupinami motorů M_1, M_2 – *Obr.4.3*, které jsou umístěné v nohách jeřábu.



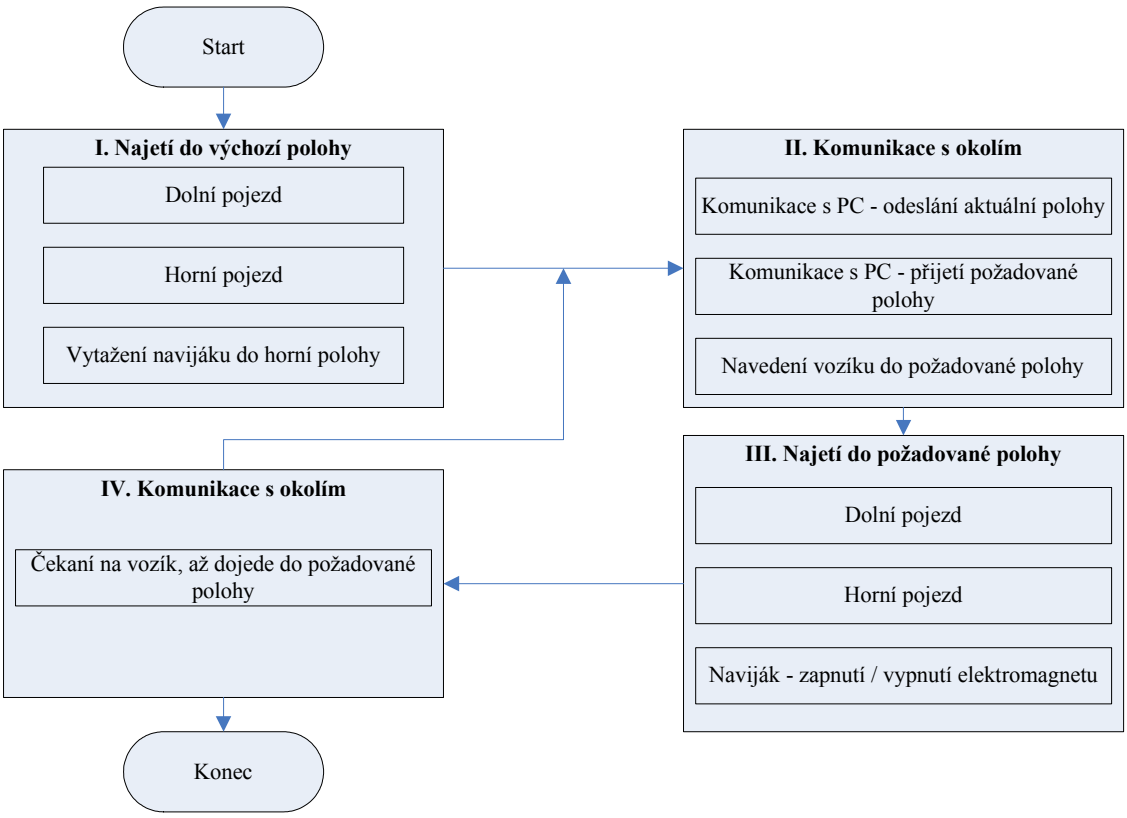
Obr.4.3 Uspořádání mechanických částí portálového jeřábu

K zabezpečení kolmosti jeřábového mostu na jízdní dráhu je možno p($M_H (v_H, s_H)$ snímače zpříčení. Při realizaci modelu portálového jeřábu bylo

použito odměřování odchylky drah $\Delta s = s_1 - s_2$ pomocí inkrementálních lineárních snímačů polohy umístěných na nohách jeřábu. Při vyskytnutí odchylky Δs se po jejím vyhodnocení zbrzdí buď motor M_1 nebo motor M_2. Součástí každého z motorků je převodovka, kde otáčky jejího výstupního hřídele jsou naprázdno 44 ot./min. Dále je převod uskutečněn pomocí řemenových převodů, kterými jsou kompenzovány nepřesnosti stavebnice Merkur.

Na každém takovémto zařízení se nachází mnoho bezpečnostních prvků, proto nemůžou chybět ani na tomto modelu. Jsou reprezentovány sedmi koncovými spínači, které se nachází na konci drah (kolejích) a na závěsném laně.

Celý model je řízen hlavním řídícím mikropočítačem, který se nachází mimo jeřáb. Chování modelu je patrné z následujícího vývojového diagramu (*Obr.4.4*). Jednotlivé kroky budou popsány dále.



Obr.4.4 Chování modelu portálového jeřábu

Stručný popis chování modelu:

I. Po spuštění systému najede nejprve celý mechanismus do nulové polohy, aby bylo vždy zaručeno přesné odměřování polohy. Celý pohyb se provádí postupně, a to v pořadí – najetí do nulové polohy dolním pojezdem, horním pojezdem a poté se provede navinutí lanka do horní polohy.

II. Když model dokončí pohyb, začne řídící mikrokontrolér vysílat svoji polohu do obsluhujícího PC. Na počítači se objeví aktuální poloha a obslužný program čeká na zadání požadované polohy, do které se má jeřáb s vozíkem přesunout. Po zadání vyšle tuto polohu program do mikrokontroléru a čeká až začne řídící mikropočítač opět vysílat. Hned po přijetí žádané polohy navede řídící mikropočítač do této polohy také vozík.

III. Poté začne samotný přesun systému do nové polohy v pořadí – dolní pojezd, horní pojezd a odvinutí popř. navinutí lana a zapnutí popř. vypnutí elektromagnetu.

IV. Když se dostane jeřáb do nové polohy, zjistí, jestli i vozík do této nové polohy dorazil, pokud ne, čeká až do jeho doražení. Pak začne opět komunikovat řídící mikrokontrolér s obsluhujícím osobním počítačem.

5. Hardwarové vybavení modelu

Celý mechanismus je napájen z počítačového zdroje s výkonem 250W a elektronické prvky se nachází na třech propojených deskách. Dvě z nich jsou umístěny do vyprázdněné počítačové skříně, ve které se nachází i již zmíněný zdroj napětí. Reset hlavního mikrokontroléru, stejně jako LED dioda indikující zapnuté napájení, jsou vyvedeny na přední část skříně. Hlavní deska je propojena konektorem s devíti vodiči s deskou pro řízení motorů, na které se nachází čtyři tranzistorové měniče (viz kapitola 5.2.). Vstupní část konektoru pro komunikaci s vozíkem je galvanicky oddělena přes optoelektrický prvek PC827. Celý tento komunikační konektor je spojen šesti vodiči s řídící deskou vozíku. Deska, která je namontována na samotném modelu jeřábu, je nepojena pomocí paralelního kabelu k hlavní řídící desce. Všechny desky mají kromě obsazených pinů mikrokontrolérů vyvedené i další nevyužité vývody mikrokontrolérů, které jsou tak připraveny na případné modifikace celého modelu.

Elektrická schémata, stejně jako návrhy plošných spojů, jsou navrženy v programu Eagle 4.11, jehož demo verze, stejně jako zdrojové soubory navržených schémat a plošných spojů, jsou na přiloženém CD. Výhodou tohoto programu jsou obsáhlé knihovny elektronických součástek, které umožňují snadné vytváření návrhů schémat, jejichž správné propojení lze v určitých vlastnostech kontrolovat přímo po navržení. Plošné spoje jsou jednostranné, což je pro jednodušší schémata použitých desek zcela dostačující. Mechanismus jeřábu je sestrojen z kovové stavebnice Merkur, která umožňuje velkou možnost modifikace celé konstrukce.

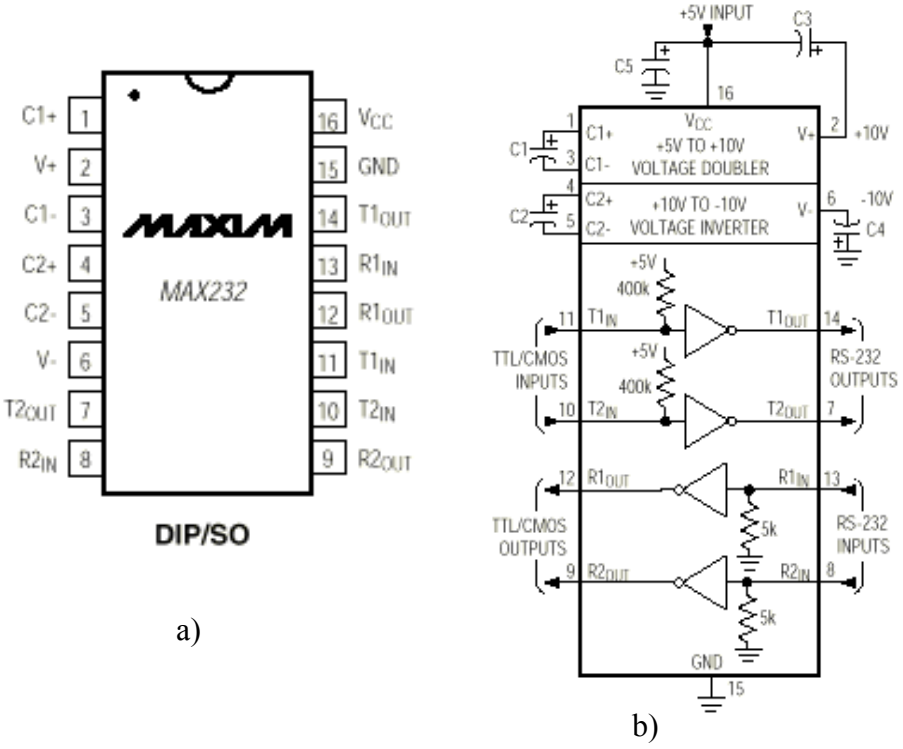
5.1 Hlavní deska

Kompletní schéma zapojení hlavní desky spolu s rozložením součástek na tištěném spoji je v Příloze č.2.

Na této desce se nachází řídící mikropočítač (PIC16F877), který je propojen s každým ze dvou mikrokotrolérů PIC16F84A pomocí třech vývodů. Všechny tři mikrokontoléry mají vnější hodinový krystal s taktem 20 MHz (HS krystal) a jejich I/O porty jsou chráněny minimálně 300Ω odpory, aby nebyl nikdy překročen maximální možný proud na jednotlivých pinech.

Aby bylo možné připojit logiku mikrokontroléru pomocí rozhraní RS232 přímo na sériový port osobního počítače, je nutné upravit napěťové úrovně této sériové linky na hodnoty uvedené v kapitole 3.5.3. K tomu se používá integrovaný obvod MAX232

firmy Maxim, který spolu s externími kondenzátory dokáže propojit tyto navzájem neslučitelné napěťové úrovně, a proto se nachází také na hlavní desce. Obvod MAX232 je převodník TTL na RS232. Obsahuje dvě dvojice oddělovačů konvertujících napěťové úrovně. Napětí pro RS232 se získává pomocí nábojové pumpy. Zapojení vývodů a aplikační schéma obvodu MAX232 je na *Obr.5.1.*



Obr 5.1 a) Zapojení vývodů a b) aplikační schéma obvodu MAX232

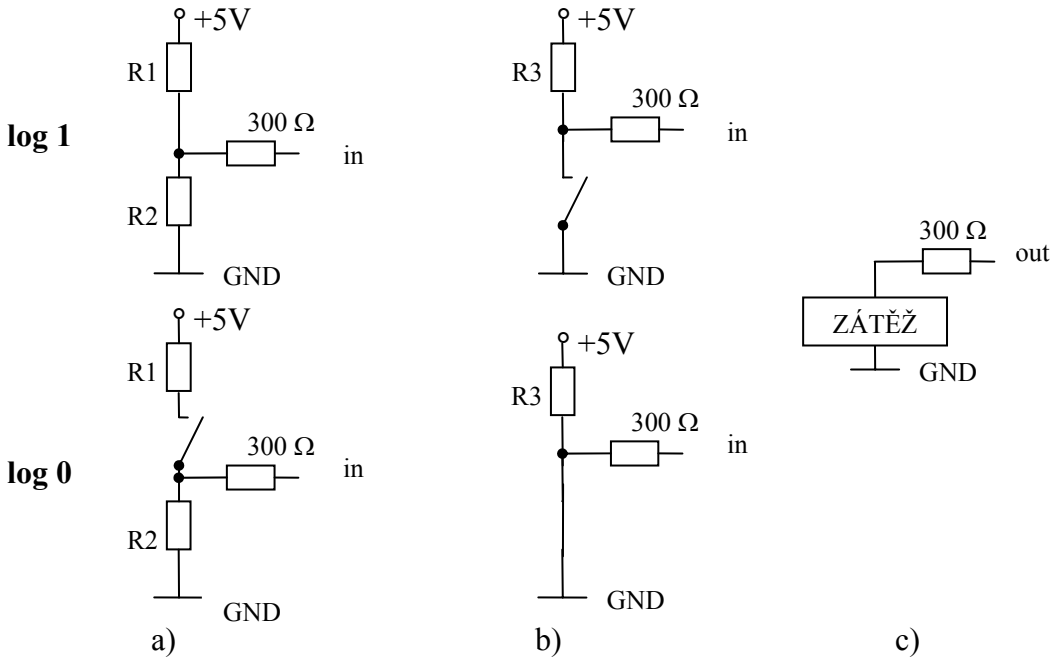
PORTA a PORTB řídícího jednočipového mikropočítače PIC16F877 je možné pomocí propojovacího kabelu a několika propojek („jumperů“) zapojit buď jako vstup (*Obr.5.2a*) nebo jako výstup (*Obr.5.2b*). To umožňuje použít hlavní desku nejen pro tuto konkrétní úlohu, ale také pro jiné studijní příklady a výuku studentů. PORTC je z velké části nevyužit a je vyveden na konektor, ke kterému lze připojit další součástky popř. sběrnici I^2C.

5.1.1 Vnější zapojení vývodů mikrokontrolérů

Je nutné dbát na to, aby každý pin nastavený jako vstupní, byl stále připojen buď k zemi nebo 5V, jinak je vstup nedefinován a stává se z něho tzv. plovoucí vstup, který nahodile mění svojí hodnotu. Při navrhování vnějšího zapojení mikrokontrolérů se musí brát na zřetel vnitřní logika portů. Vnitřní zapojení jednotlivých pinů může být buď

zapojeno jako tzv. Schmittův klopný obvod nebo jako TTL logika. Z *Obr.5.2a* a *Obr.5.2b* jsou patrny vnější zapojení na pinech určených pro vstup. Výstup je u obou logik stejný a je znázorněn na *Obr.5.2c*. Vnitřní logika portů v základním nastavení, které je uskutečněno hned po resetu nastavením některých SFR, především registrů, které vypnou analogové vstupy, u mikrokontrolérů PIC16F877 a PIC16F84A je v Příloze č.1.

Odpor 300Ω je stejný u vstupu i výstupu a je určen k ochraně vývodu mikrokontroléru, je tím zabezpečeno, že nebude překročen maximální proud na pinech(25 mA). Odpory R1 a R2 při TTL logice mohou být v rozsahu 0,5kΩ až 2,5kΩ, čímž je zaručeno, že při změně vstupu bude mikrokontrolér reagovat. Pro realizaci modelu jsou zvoleny odpory R1 = R2 = 1kΩ. Tyto odpory tvoří tzv. napěťový dělič a jelikož jsou si rovny, je napětí na vstupu cca 2,5 V, což je dostačující úroveň napětí pro rozlišení logické 1 a logické 0. Odpor R3 je nastaven na 10kΩ.



Obr.5.2 a) vstup TTL logika (pozitivní logika) b) vstup ST logika (negativní logika) c) výstup obě logiky

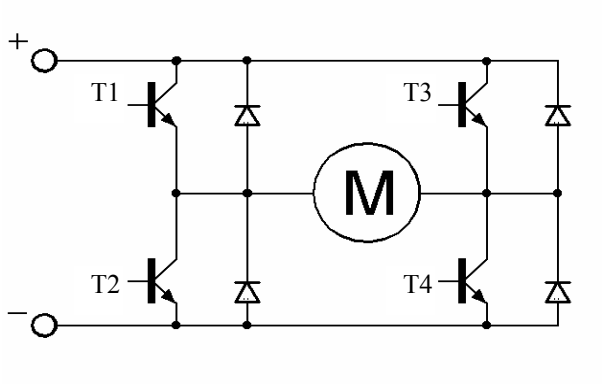
5.2 Řídící deska motorů

Na této desce se nachází elektronické zapojení, které umožňuje řídit směr a otáčení jednotlivých motorů. Deska má na sobě umístěny dva konektory, k prvnímu je připojena hlavní deska, konkrétně výstupy z mikrokontrolérů PIC16F84A, a druhým

jsou napojeny zpět přes hlavní desku a jeřáb jednotlivé motory. Kompletní schéma zapojení této řídící desky motorů spolu s návrhem plošného spoje a jeho osazením součástkami je v Příloze č.2.

5.2.1 Řízení směru otáčení motorů

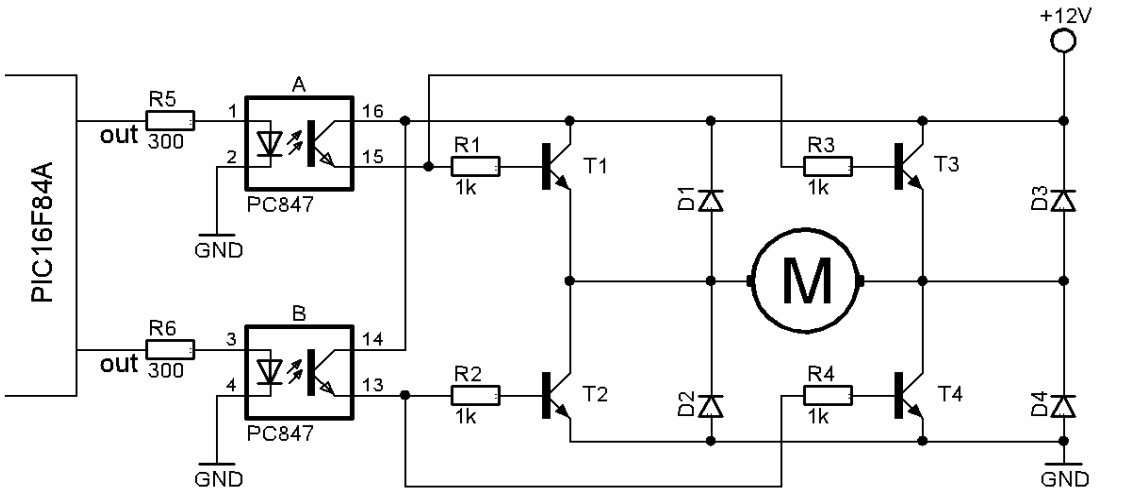
Princip řízení směru otáčení motorů vychází z tranzistorového měniče. Konkrétně ze čtyřkvadrantového tranzistorového stejnosměrného pulzního měniče, který je schopen dávat na výstupu oba směry proudu a obě polarity napětí. Tranzistorový pulzní měnič je měnič napěťový, tzn. je napájen ze zdroje napětí. Schéma tranzistorového pulzního měniče určeného pro pohon stejnosměrného servopohonu je na *Obr.5.3*. Diody zapojené mezi emitorem a kolektorem u jednotlivých tranzistorů jsou ochranné a zamezují zničení tranzistorů napěťovými překmity vznikajícími na motoru.



Obr.5.3 Schéma tranzistorového měniče

Pro pohon motorů modelu portálového jeřábu je schéma měniče zcela zachováno, jsou v něm použity tranzistory typu NPN s označením BD139-16 a polovodičové diody 1N4007. Všechny čtyři motory ovládají dva mikrokontoléry PIC16F84A. Jeden vývod mikrokontroléru ovládá vždy dva tranzistory (T1 a T4 nebo T2 a T3) měniče jednoho motoru.

Pokud se pro jeden motor nastaví příslušný vývod na log 1 a tím se sepnou tranzistory T1 a T4, musí být vývod ovládající tranzistory T2 a T3 nastaven na log 0. V případě, že by oba z dvojice vývodů ovládajících jeden motor byly nastaveny na log 1, došlo by k poškození obvodu. Tomuto je zamezeno softwarově, vše je patrné z Přílohy č.3 a 4(programy mikrokontrolérů PIC16F84A). Mohou tedy nastat pouze stavy 00, 01 a 10. Vývody těchto mikrokontrolérů jsou galvanicky oddělené pomocí optočlenů PC847, a to nejen z důvodu ochrany, ale také protože mikrokontrolér je napájen 5V, kdežto motory jsou napájeny 12V. Spínací proud báze, a tím i odpor před bází, záleží na velikosti zátěže, kterou je nutné spínat. Pro použité motory, které odebírají maximální proud 300mA a průměrně odebírají proud cca 100mA, vyhovuje odpor 1kΩ. Celé schéma řízení jednoho motoru je znázorněno na *Obr.5.4*.



Obr.5.4 Schéma řízení pro jeden motor

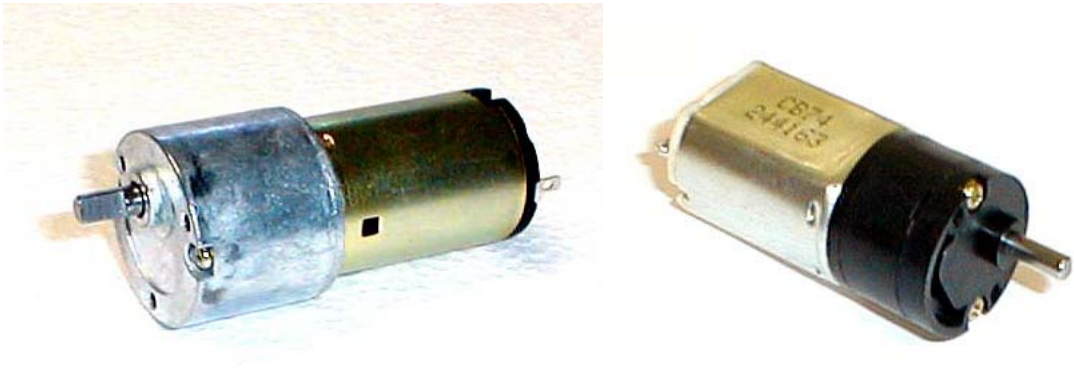
5.3 Propojovací deska a součásti umístěné na jeřábu

Na této třetí desce, která se nachází na boční straně levé nohy modelu jeřábu, jsou připojeny samotné motory, snímače a koncová čidla. Je na ní také umístěn konektor paralelního portu, který slouží k připojení kabelu vedoucího z hlavní řídící desky.

5.3.1 Pohonné jednotky

K pohonu modelu portálového jeřábu slouží čtyři stejnosměrné modelářské elektromotory firmy MEGA MOTOR s převodovkami integrovanými již od výrobce. K pohybu dolního pojezdu jsou použity větší, a tudíž i silnější elektromotory s označením G33 / N2738, aby bylo zaručeno, že jejich moment utáhne celý mechanismus jeřábu. Jako pohon horního pojezdu a pohonu navijáku slouží menší verze elektromotoru s označením 20GN152025. Oba motory jsou zobrazeny na *Obr.5.5*.

Rozměry jednotlivých motorů spolu s jejich hlavními vlastnostmi jsou v *Tabulce.5.1*.



a) b)

Obr.5.5 Elektromotor s převodovkou firmy MEGA MOTOR
a) G33 / N2738 b) 20GN152025

		Elektromotor s převodovkou	
		G33 / N2738	**20GN152025**
Rozměry	Délka motoru **L** [mm]	35,0	25,0
	Délka převodovky **l** [mm]	24,5	14,4
	Průměr převodovky **D** [mm]	33,0	20,0
	Průměr hřídele **d** [mm]	5,0	3,5
	Hmotnost **m** [g]	125	30
	Napájecí napětí **U** [V]	6 - 15	4 - 12
	Převodový poměr **i**	125	380
	Otáčky naprázdno $\mathbf{n_0}$[ot/min]	44	25
Max. zatížení	Radiální [N]	50	2,5
	Axiální [N]	33	2
Bod max. účinnosti	Otáčky **n** [ot/min]	38	23
	Proud **I** [A]	0,3	0,08
	Moment **M** [mNm]	300	60

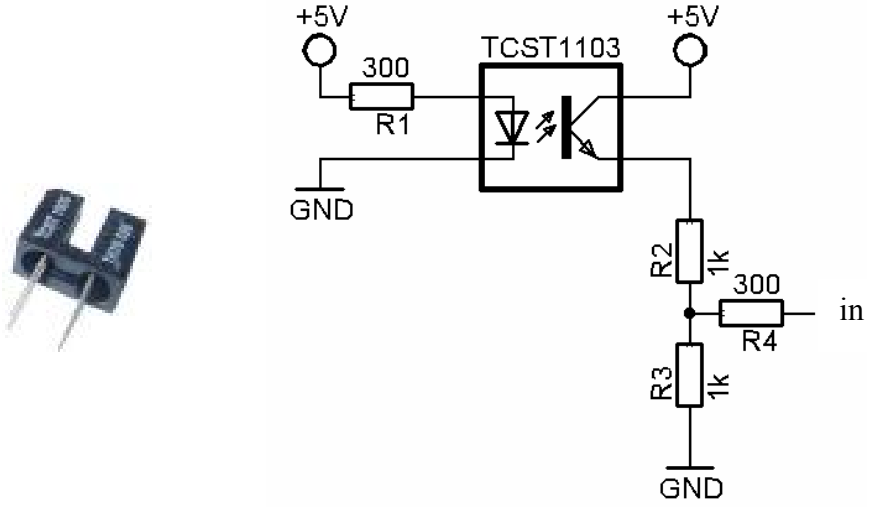
Tabulka 5.1 Rozměry a vlastnosti použitých elektromotorů

5.3.2 Inkrementální snímače a koncová čidla

Pro určení aktuální polohy dolního a horního pojezdu jsou použity inkrementální lineární snímače. Jedná se o štěrbinové světelné závory s označením TCST 1103, které se skládají z diody a fototranzistoru. Jako clona je použita děrovaná kolejnice ze stavebnice Merkur, která vede u dolního pojezdu podél pojezdové kolejnice a u horního pojezdu je sama pojezdovou kolejnicí. Rozteč mezi jednotlivými dírami clony je 1cm a jejich průměr je 4 mm. Schéma zapojení závory a její podoba jsou znázorněny na *Obr.5.5*.

Celé odčítaní je cyklické, což znamená, že stav fototranzistoru se odčítá stále dokola s tím, že se uchovává předchozí stav, který se následně vyhodnocuje. Při realizaci se vyskytly při přechodech stavů z log 1 na log 0 a naopak zákmity, které bylo nutné vyřešit. Jako nejjednodušší způsob se ukázalo softwarové řešení (Příloha č.3 – funkce KMIT), které opakuje čtení stavu fototanzistoru několikrát za sebou a v případě, že po určenou dobu je stav stejný, přistoupí se k vyhodnocení předchozího stavu. Pokud je předchozí stav log 0 vzroste počet o jedničku.

Jako koncová čidla jsou použity malé mechanické spínače. Čtyři z těchto spínačů jsou napojeny na porty RB4 až RB7 mikrokontroléru PIC16F877, které umožňují při změně stavu vyvolat přerušení, a tím rychle reagovat ve chvíli, kdy se jeřáb dostane do krajní polohy. Po vyvolání přerušení a určení od jakého vývodu toto přerušení přišlo, najede jeřáb do nulové nebo maximální polohy. Jednotlivé funkce jsou v Příloze č.3 a jejich název začíná na ODJETI.



Obr.5.5 Schéma zapojení inkrementálního čidla

6. Programová část modelu

Pro vývoj jednotlivých programů určených pro mikrokontroléry bylo použito vývojové prostředí MPLAB verze 7.01, které je volně k dispozici na webových stránkách společnosti Microchip. Tento program umožňuje kromě samotného napsání programového kódu také simulaci, a tím se stává opravdu velmi užitečným nástrojem při programování jednočipových mikropočítačů. Další velkou výhodou je vytvoření zdrojového kódu do samostatných celků, které poté lze pomocí direktivy „include“ vkládat do hlavního programu. K tomu také slouží tzv. makra, která jsou v podstatě šablony s parametry. Tím se nejen zpřehledňuje celý program, ale také je možné tyto celky použít i pro jiné programy.

Mikrokontroléry PIC umožňují nastavit vlastnosti jednotlivých portů během programu, ale jejich základní nastavení se provádí při tzv. inicializaci, která je spuštěna vždy po každém resetu daného mikropočítače. Při tomto nastavení se určí, který port bude vstupní a který výstupní popř. který vývod portu, a také se nastaví potřebné SFR (speciální funkční registry).

6.1 Nastavení a funkce jednotlivých portů mikrokontrolérů

Hlavní řídící mikrokontrolér PIC16F877

PORTA : Celý port je nastaven jako vstupní.

- RA0 : Signál z inkrementálního čidla, které odčítá ujetou dráhu na levé kolejnici.
- RA1: Signál z inkrementálního čidla, které odčítá ujetou dráhu na pravé kolejnici.
- RA2 : Signál z inkrementálního čidla, které odčítá ujetou dráhu horního pojezdu.

PORTB : Celý port je nastaven jako vstupní.

- RB1 : Indikuje převití navijáku.
- RB2 : Koncové čidlo umístěné na pravé kolejnici v maximální vzdálenosti.
- RB3 : Koncové čidlo umístěné na pravé kolejnici v nulové vzdálenosti.
- RB4 : Koncové čidlo umístěné na levé kolejnici v maximální vzdálenosti.
- RB5 : Koncové čidlo umístěné na levé kolejnici v nulové vzdálenosti.
- RB6 : Koncové čidlo umístěné na horním pojezdu nad levou dolní kolejnicí.
- RB7 : Koncové čidlo umístěné na horním pojezdu nad pravou dolní kolejnicí.

PORTC : Vývody tohoto portu jsou používány jako vstupní i jako výstupní.

- RC2 : Nastaven jako výstupní, zapíná a vypíná elektromagnet.
- RC6 : Vývod slouží jako TxD (vysílací pin) pro asynchronní komunikaci.
- RC7 : Vývod slouží jako RxD (přijímací pin) pro asynchronní komunikaci.

PORTD : Tento port je nastaven celý jako výstupní.

- RD0 : Slouží pro komunikaci s vozíkem.
- RD1 : Slouží pro komunikaci s vozíkem.
- RD2 : Určuje směr otáčení motorů horního pojezdu a navijáku. Je připojen na první PIC16F84A.
- RD3 : Určuje stav navíjecího motoru(1 – jeď, 0 – stůj). Je připojen na první PIC16F84A.
- RD4 : Určuje stav motoru horního pojezdu(1 – jeď, 0 – stůj). Je připojen na první PIC16F84A.
- RD5 : Určuje směr otáčení motorů dolního pojezdu. Je připojen na druhý PIC16F84A.
- RD6 : Určuje stav levého motoru dolního pojezdu (1 – jeď, 0 – stůj). Je připojen na druhý PIC16F84A.
- RD7 : Určuje stav pravého motoru dolního pojezdu (1 – jeď, 0 – stůj). Je připojen na druhý PIC16F84A.

PORTE : Celý port je nastaven jako vstupní.

- RE0 : Slouží pro komunikaci s vozíkem.
- RE1 : Slouží pro komunikaci s vozíkem.

Pomocné mikrokontroléry pro ovládání motorů PIC16F84A

PORTA : Tento port je nastaven jako vstupní.

- RA1 : Je připojen na vývod RD2, resp.RD5 řídícího mikrokontroléru.
- RA2 : Je připojen na vývod RD3, resp.RD6 řídícího mikrokontroléru.
- RA3 : Je připojen na vývod RD4, resp.RD7 řídícího mikrokontroléru.

PORTB : Celý port je nastaven jako výstupní.

- RB4 : Ovládá tranzistory řídící jednotky pro levý motor dolního pojezdu, resp. motor horního pojezdu pro jízdu vpřed.
- RB5: Ovládá tranzistory řídící jednotky pro levý motor dolního pojezdu, resp. motor horního pojezdu pro jízdu vzad.

- RB6 : Ovládá tranzistory řídící jednotky pro pravý motor dolního pojezdu, resp. motor navijáku pro jízdu vpřed, resp. odvíjení.
- RB7 : Ovládá tranzistory řídící jednotky pro pravý motor dolního pojezdu, resp. motor navijáku pro jízdu vzad, resp. navíjení.

Ostatní vývody jednotlivých portů u všech mikrokontrolérů jsou vyvedeny na rezervní konektory, a tím je dána možnost rozšíření modelu nebo případné modifikace celé úlohy.

6.2 Zdrojový kód PIC16F877

Zdrojový kód určený pro hlavní řídící mikrokontrolér je v Příloze č.3. V první části kódu jsou nadefinovány potřebné proměnné (registry) a jejich adresy v paměti, pomocí direktivy „define“ přehledněji označeny jednotlivé vývody portů a nastaveno konfigurační slovo, ve kterém je definováno základní nastavení mikrokontroléru. Dále je také nastavené přerušení od vývodů RB4 – RB7. Od návěstí MAIN začíná hlavní program, ze kterého jsou postupně volány obsluhující funkce. Vývojové diagramy, které pomohou k lepší orientaci v řešení jsou v Příloze č.4.

Komunikace s osobním počítačem pomocí rozhraní RS232 je nastavena na rychlost 19200 Bd a je nastaven 8-bitový přenos bez parity, která není nutná, protože přenos probíhá na krátkou vzdálenost. Při této komunikaci vyšle mikrokontrolér postupně čtyři osmibitová čísla, která reprezentují aktuální polohu systému. Poté čeká na požadovanou polohu poslanou z PC, která přichází také v podobě čtyř osmibitových čísel. Navedení vozíku na požadovanou polohu je provedeno nastavením kombinace vývodů RD0 a RD1. Naopak, zjišťování jeho aktuální polohy je provedeno čtením PORTuE.

Hlavní obsluhující funkce:

- **JIZDAVLEVO, JIZAVPRAVO** – Tyto dvě funkce obsluhují jízdu dolního pojezdu. Porovnáním aktuální polohy (OTACKYL) a polohy žádané (OTACKYLPPC) se rozhodne, která z těchto dvou funkcí se použije. Jejich rozdíl je především v tom, jestli je hodnota v registru OTACKYL resp.OTACKYP zmenšována nebo zvětšována. V těchto funkcích se také porovnávají hodnoty v registrech OTACKYL a OTACKYP a v případě jejich nerovnosti se nastaví RD6 resp. RD7 na log 0 a tím se zabrzdí motor M1 resp.

M2. Tímto je vyřešen problém se zpříčením jeřábového mostu. Při rovnosti OTACKYL a OTACKYLPPC, se pohyb dolního pojezdu zastaví a program se vrátí do jeho hlavní části.

- **JIZDAVLEVOHORNI, JIZAVPRAVOHORNI –** Tyto funkce se starají o pohyb horního pojezdu. Podobně jako předchozí dvě funkce, také mezi těmito se rozhoduje porovnáním obsahů dvou registrů a OTACKYH a OTACKYHPC, který reprezentují aktuální a požadovanou polohu horního pojezdu. Jelikož se při pohybu horního pojezdu nekontroluje žádné zpříčení, kontroluje se zde pouze nerovnost hodnot v registrech OTACKYH a OTACKYHPC, při jejich rovnosti se motor horního pojezdu zastaví a hlavní program pokračuje dál.
- **NAVIJODVIJ** – Tato funkce se stará o odvíjení a následné nevíjení lana navijáku a o zapnutí resp. vypnutí elektromagnetu. Doba navíjení resp. odvíjení je pevně daná a je nastavena na 5 sekund.
- **TIME** – Volání této funkce se objevuje během celého programu vícekrát. Reprezentuje časovou smyčku s délkou 1 sekunda, která je do zdrojového kódu vkládána z důvodu oddělení jednotlivých pohybů jeřábu.
- **ODJETIDOLNILEVE, ODJETIDOLNIPRAVE, ODJETIHORNILEVE, ODJETIHORNIPRAVE** – Tyto funkce nejsou volány přímo z hlavního programu, ale jsou volány při přerušení. Mají na starosti najetí do nulové resp. maximálně možné polohy dolního resp. horního pojezdu v případě, že jsou sepnuty koncové snímače, čímž je detekováno, že se mechanizmus dostal do krajní polohy. Vzdálenost nulové polohy je od krajní polohy rovna vzdálenosti, kterou ujede pojezd za dobu 0,5 sekundy.

6.3 Programový kód PIC 16F84A

Zdrojový kód pro oba mikrokontroléry PIC16F84A je stejný a slouží k ovládání otáček jednotlivých motorů a směru jejich otáčení pomocí pulzní šířkové modulace. Tento kód je v Příloze č.5 a vývojové diagramy, k těmto kódům v Příloze č.6. První z těchto mikrokontrolérů ovládá motory dolního pojezdu a druhý motor horního pojezdu a motor určený k pohonu navijáku. Povely jdou z řídícího mikropočítače, a to nastavením logiky třech vývodů, při jejichž kombinaci se nastavují čtyři vývody mikrokontrolérů PIC16F84A, které jsou napojeny na tranzistorové měniče. Nastavení této kombinační logiky je uvedeno v *Tabulce 6.1*.

Na začátku zdrojového kódu je nastaveno konfigurační slovo, adresy registrů použitých pro proměnné a pojmenovány jednotlivé vstupy a výstupy pro lepší přehlednost. V hlavní části programu jsou postupně volány čtyři obslužné funkce. Aby byla zaručena rovnoměrná pulzní šířková modulace, je na konci hlavního programu čekací smyčka, která trvá do doby, než je detekováno přetečení registru TMR0.

Pulzní šířková modulace je reprezentována nastavováním výstupních vývodů na log1 a log0 podle střídy. Střídou je zde číslo, které v případě, že je menší než pět, způsobí nastavení vývodu na log0, v opačném případě na log1. Maximálně může být toto číslo rovno 40. Při každém průchodu hlavní smyčkou, je toto číslo sníženo o jedničku. V Případě, že se rovná nule, je opět nastaveno na aktuální hodnotu střídy.

Vstupy			**Výstupy**				
			1. Motor		**2.Motor**		
RA1	RA2	RA3	RB4	RB5	RB6	RB7	**Poznámky**
1	1	1	1	0	1	0	Oba motory jedou vlevo
1	0	1	1	0	0	0	První motor jede vlevo druhý stojí.
1	1	0	0	0	1	0	První motor stojí druhý jede vlevo.
0	1	1	0	1	0	1	Oba motory jedou vpravo.
0	0	1	0	1	0	0	První motor jede vpravo druhý stojí.
0	1	0	0	0	0	1	První motor stojí druhý jede vpravo.
0	0	0	0	0	0	0	Oba motory stojí.
1	0	0	0	0	0	0	Oba motory stojí.

Tabulka 6.1 Nastavení kombinační logiky pro vývody RB4-RB7

6.4 Aplikace pro ovládání modelu na PC

Důležitou součástí celého modelu systému pro přepravu materiálu je ovládání pomocí osobního počítače, kam lze přes jednoduché grafické rozhraní zadávat požadovaný pohyb jeřábu a vozíku. Také je zde schématicky znázorněn aktuální stav celého systému, který je aktualizován vždy po skončení pohybu jeřábu a vozíku.

Aplikace je napsána v prostředí Borland C++ Builder. Toto prostředí je v dnešní době jedním z nejpoužívanějších systémů pro tvorbu aplikací na PC. Toto prostředí se vyznačuje některými specifickými prvky, které programátorovi významně zjednodušují vývoj aplikace a zároveň umožňují vizualizaci na vysoké úrovni pomocí vestavěných komponent.

Prostředí Borland C++ Builder se vyznačuje těmito výhodami:

- maximálně zjednodušená návrhová fáze při vytváření vizuálního prostředí aplikace s využitím vlastností operačního systému
- velké množství integrovaných vývojových nástrojů a pomůcek a rovněž velké množství volně dostupných již hotových komponent
- objektově orientovaný přístup
- velmi dobře propracovaná podpora databází

Vytvořená aplikace s názvem ModelMove poskytuje následující funkce:

- vybrání komunikačního COM portu
- otevření a uzavření COM portu
- nastavení komunikační rychlosti
- nastavení paritního nebo bezparitního přenosu
- nastavení délky Stop bitu a přenášeného slova
- grafické i číselné zobrazení aktuální polohy celého systému
- zadání požadované polohy celého systému

Popis obrazovky vytvořené aplikace ModelMove

Hlavní okno aplikace ModelMove je pro přehlednost rozděleno do tří logických bloků :

1. V prvním bloku, který se nachází v horní půlce okna, lze vybrat komunikační port COM1 až COM8. Lze volit mezi rychlostmi komunikace , délkou Stop bitu, délkou přenášeného slova a také lze zvolit paritu přenosu.
2. V dolní polovině okna v záložce Terminál, je vyobrazené terminálové okno, ve kterém jsou vidět přicházející a odcházející data.
3. Poslední blok, který se nachází v záložce Jeřáb, slouží k samotnému ovládání modelu. Je zde znázorněna aktuální poloha celého systému jak číselně, tak v grafické podobě. Také se zde zadává požadovaná poloha, do které se má model přesunout.

Spouštěcí soubor ModelMove.exe této aplikace se nachází na přiloženém CD.

7. Závěr

Cílem této diplomové práce bylo vytvoření didaktické pomůcky, která by pomohla budoucím studentům lépe pochopit principy servomechanismů a jejich řízení pomocí jednočipových mikropočítačů. K tomuto účelu byl sestrojen model systému jeřábu a vozíku, který je určen k přepravě materiálu z jednoho místa na druhé.

V části, určené pro pochopení celého problému, byly popsány principy servopohonů a jejich základní rozdělení. Dále zde byla přiblížena problematika jednočipových mikropočítačů, především mikrokontrolérů PIC, které vyrábí firma Microchip. Důležitou částí je také popis komunikačních možností mikrokontrolérů a jejich propojení s jinými logickými obvody nebo osobním počítačem.

Samotnému návrhu modelu jeřábu a jeho sestrojení se věnují kapitoly čtyři a pět. V nich jsou postupně ukázány principy chování modelu, jeho uspořádání jak z pohledu konstrukce, tak elektrického zapojení, a v neposlední řadě také programového vybavení všech mikrokontrolérů. Šestá kapitola seznamuje s aplikací určenou pro osobní počítač, pomocí které je nejen ovládán pohyb celého systému, ale také je schématicky znázorněna jeho aktuální pozice.

Celý model portálového jeřábu nelze považovat za konečný. Mechanizmus jeřábu, stejně jako jeho ovládací prvky, umožňují svou stavbou budoucí modifikaci a rozšíření. Komunikace s vozíkem by mohla být v budoucnu provedena pomocí sběrnice I^2C, což by umožnilo připojení více integrovaných obvodů, které by mohly být použity pro ovládání dalších autonomních systémů. Pro lepší možnosti ovládání a jeho zpřesnění by bylo možné použít jiných metod odměřování polohy a kontroly zpříčení jeřábového mostu. Další oblastní pro zlepšovaní je aplikace běžící na PC, kterou je možné rozšířit o další moduly, jako je například ukládání všech poloh, kterými celý systém postupně prochází. Tato data by potom mohla sloužit pro další vyhodnocování a zpracovávání do grafů a tabulek.

Použitá literatura

[1] Souček, P. : Servomechanismy NC strojů a průmyslových robotů, Praha, ČVUT 1992

[2] Skalický, J.. : Elektrické servopohony, Brno, VUT 1999

[3] Pavelka, P.- Čeřovský, Z.- Javůrek,J. : Elektrické pohony, Praha, ČVUT 1992

[4] Vacek, V. : Učebnice programování PIC, Praha, BEN-Technická literatura 2002

[5] Hrbáček, J. : Komunikace mikrokontroléru s okolím 1, Praha, BEN-Technická literatura 2002

[6] Kalaš,V.- Jurišica, L.- Žalman, M. : Technická kybernetika elektických pohonov, Alfa Bratislava 1978

[7] Katalogové listy firmy MICROCHIP

[8] http://www.cmail.cz/doveda/

[9] http://home.zcu.cz/%7Edudacek/Pot/mikrokontrolery.pdf

[10] http://www.hw.cz

Přílohy – seznam

Příloha č.1 : Popis jednotlivých vývodů mikrokontrolérů (1 strana)

Příloha č.2 : Podrobná schémata, rozložení a seznam součástek a plošný spoj:
Hlavní desky, Řídící desky motorů a Propojovací desky (7 stran)

Příloha č.3 : Zdrojové texty souborů pro řídící mikrokontrolér PIC16F877 (15 stran)

Příloha č.4 : Vývojové diagramy ke zdrojovým textům v Příloze č.3 (6 stran)

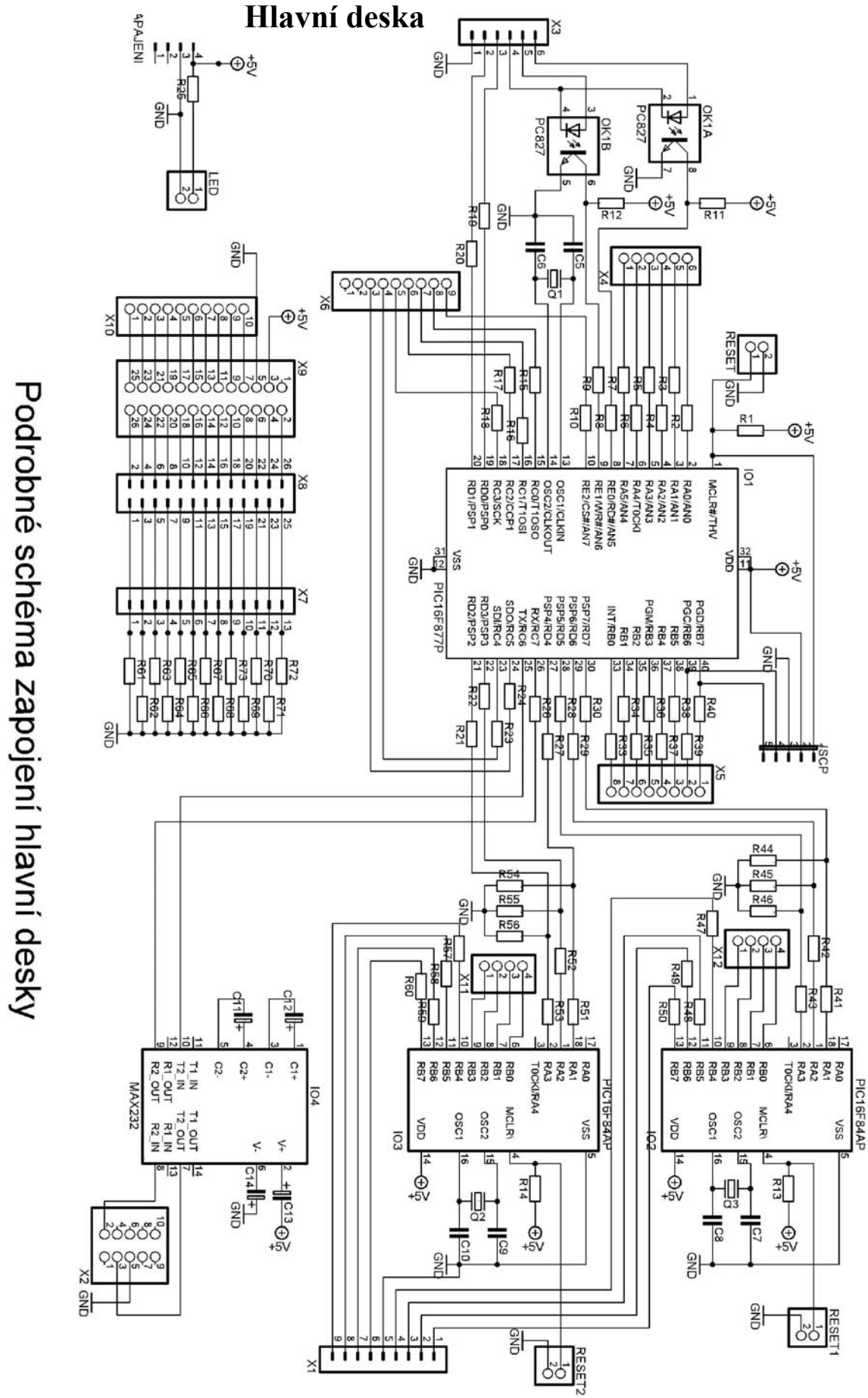
Příloha č.5 : Zdrojové texty souborů pomocných mikrokontrolérů PIC16F84A (6 stran)

Příloha č.6 : Vývojové diagramy ke zdrojovým textům v Příloze č.5 (4strany)

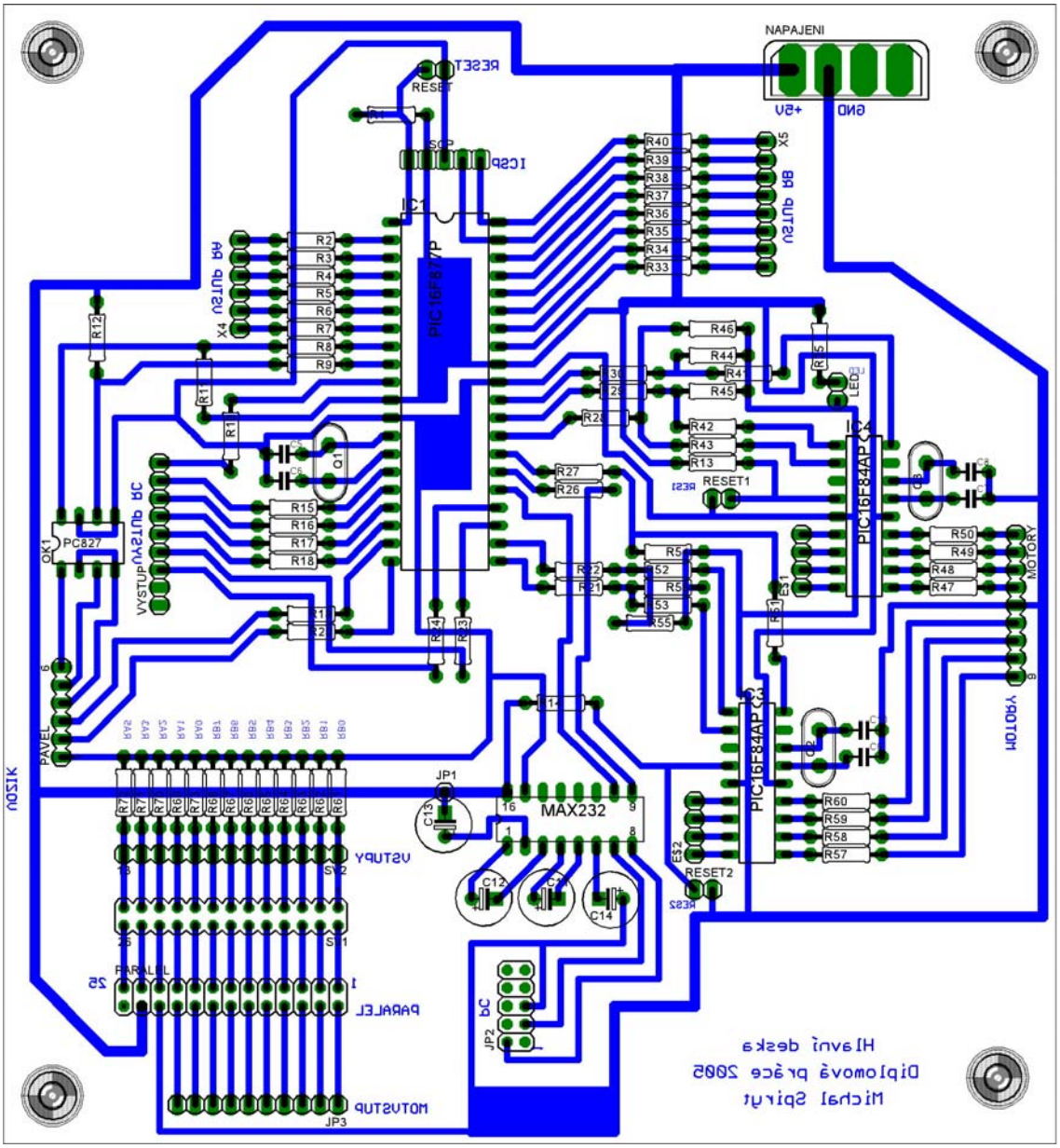
Popis jednotlivých vývodů mikrokontrolérů

vývod	Pin 16F877	Pin 16F84A	typ I/O/P	logika	popis
V_{ss}	12,31	5	P	-	zem
V_{dd}	11,32	14	P	-	napájení +5V
MCLR/V_{pp}/THV*	1	4	I/P	ST	RESET/vstup programovacího napětí. Tento vývod je aktivní v nule, kdy provádí RESET obvodu.
OSC1/CLKIN	13	16	I	CMOS	Vstup pro krystalový oscilátor/vstup ext. hodin
OSC2/CLKOUT	14	15	O	-	Výstup krystalového oscilátoru. Připojení krystalu nebo keramického rezonátoru. V RC módu výstup CLK signálu, který je 1/4 kmitočtu na OSC1
RA0/AN0*	2	17	I/O	TTL	analogový vstup 0
RA1/AN1*	3	18	I/O	TTL	analogový vstup 1
RA2/AN2*/Vref-*	4	1	I/O	TTL	analogový vstup 2/- reference
RA3/AN3*/Vref+*	5	2	I/O	TTL	analogový vstup 3/+ reference
RA4/TOCKI	6	3	I/O	ST	RA4 může být jako zdroj CLK signálu pro TMR1.
RA5/AN4/SS	7	X	I/O	TTL	analogový vstup 4
RB0/INT	33	6	I/O	TTL/ST	může být vybrán jako zdroj vnějšího přerušení
RB1	34	7	I/O	TTL	
RB2	35	8	I/O	TTL	
RB3	36	9	I/O	TTL	
RB4	37	10	I/O	TTL	přerušení při změně vstupu
RB5	38	11	I/O	TTL	přerušení při změně vstupu
RB6	39	12	I/O	TTL/ST	přerušení při změně vstupu/CLK při programování
RB7	40	13	I/O	TTL/ST	přerušení při změně vstupu/DATA při programování
RC0/T1OSC/T1CKL	15	X	I/O	ST	výstup oscilátoru nebo vstup hodin TIMER1
RC1/T1OSC/CCP2	16	X	I/O	ST	vstup oscilátoru TIMER1
RC2/CCP1	17	X	I/O	ST	
RC3/SCK/SCL	18	X	I/O	ST	hodinový vstup/výstup pro SPI a I2C
RC4/SDI/SDA	23	X	I/O	ST	datový vstup SPI nebo datový vstup/výstup I2C
RC5/SDO	24	X	I/O	ST	datový výstup SPI
RC6/TX/CK	25	X	I/O	ST	synchroní hodiny nebo USART
RC7/RX/DT	26	X	I/O	ST	synchroní data nebo USART
RD0/SP0	19	X	I/O	TTL/ST	I/O pin nebo paralel slave port bit0
RD1/SP1	20	X	I/O	TTL/ST	I/O pin nebo paralel slave port bit1
RD2/SP2	21	X	I/O	TTL/ST	I/O pin nebo paralel slave port bit2
RD3/SP3	22	X	I/O	TTL/ST	I/O pin nebo paralel slave port bit3
RD4/SP4	27	X	I/O	TTL/ST	I/O pin nebo paralel slave port bit4
RD5/SP5	28	X	I/O	TTL/ST	I/O pin nebo paralel slave port bit5
RD6/SP6	29	X	I/O	TTL/ST	I/O pin nebo paralel slave port bit6
RD7/SP7	30	X	I/O	TTL/ST	I/O pin nebo paralel slave port bit7
RE0/AN5	8	X	I/O	TTL/ST	I/O pin nebo kontrola čtení vstupu PSP nebo analog. I
RE1/AN6	9	X	I/O	TTL/ST	I/O pin nebo kontrola zápisu vstupu PSP nebo analog. I
RE2/AN7	10	X	I/O	TTL/ST	I/O pin nebo analogový I

I = Input (vstup) O = Output (výstup) I/O = Input/Output (vstup/výstup) P = Power
- = nevyužito TTL = TTL input ST = Schmittův klopný obvod * = pouze u 16F877
X = mikrokontrolér tento pin nemá



Obr.P2.1 Podrobné schéma zapojení – Hlavní deska



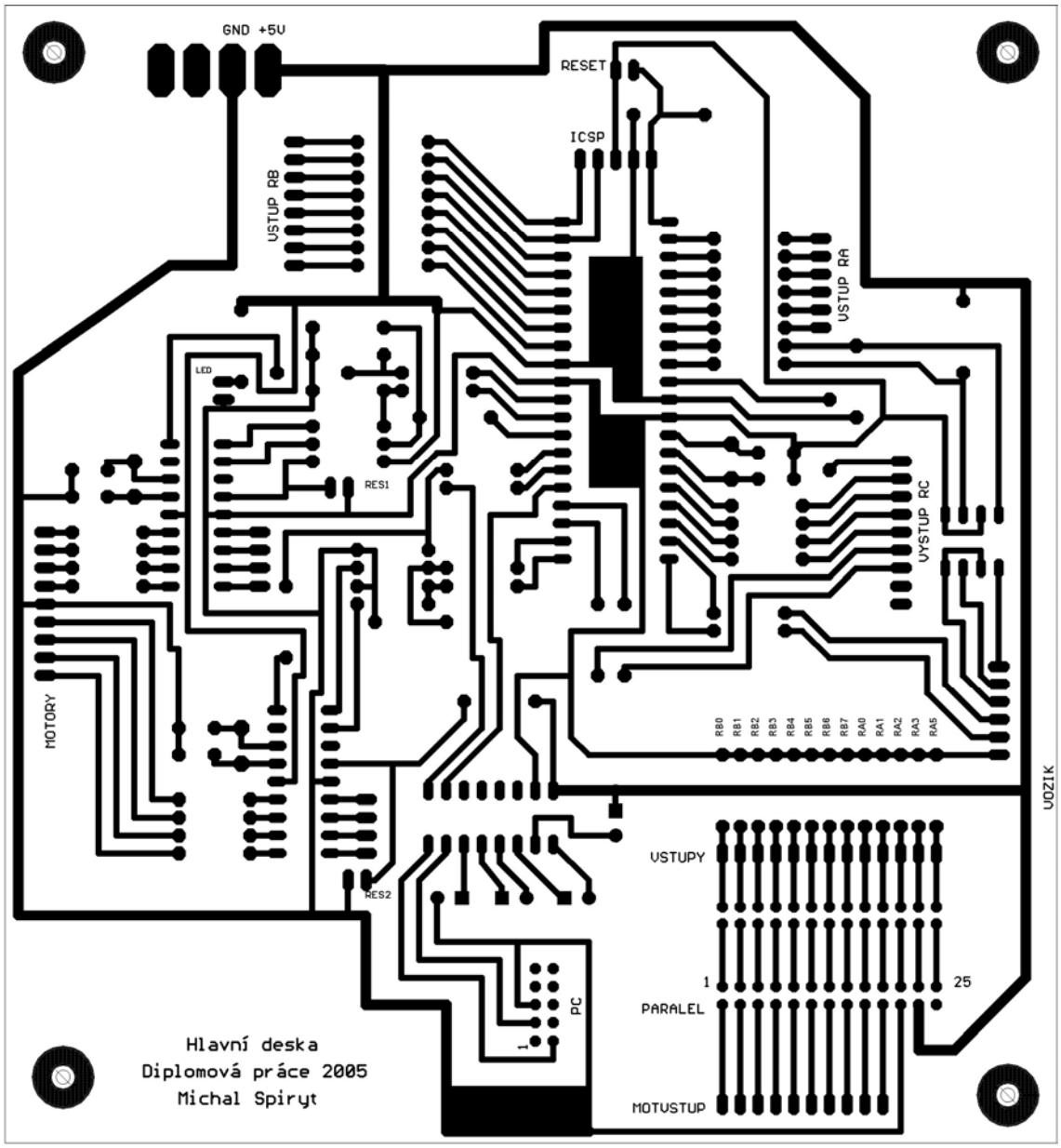
Obr.P2.2 Rozložení součástek na plošném spoji – Hlavní deska

Tabulka P2.1 Seznam součástek – Hlavní deska

R2 – R10, R15 – R20, R23, R24, R33 – R43, R47 – R53, R57 – R60	300R
R21, R22, R25, R27 – R30,R44 – R46, R54 – R56, R61 – R72	1k
R1, R11 – R14	10k
R26	820R
Q1 – Q2	20 MHz (HS)
C5 – C10	22 pF
C11 – C14	1µF
OK1	PC827
IO1	PIC16F877P/20
IO2, IO3	PIC16F84A/20
IO4	MAX232

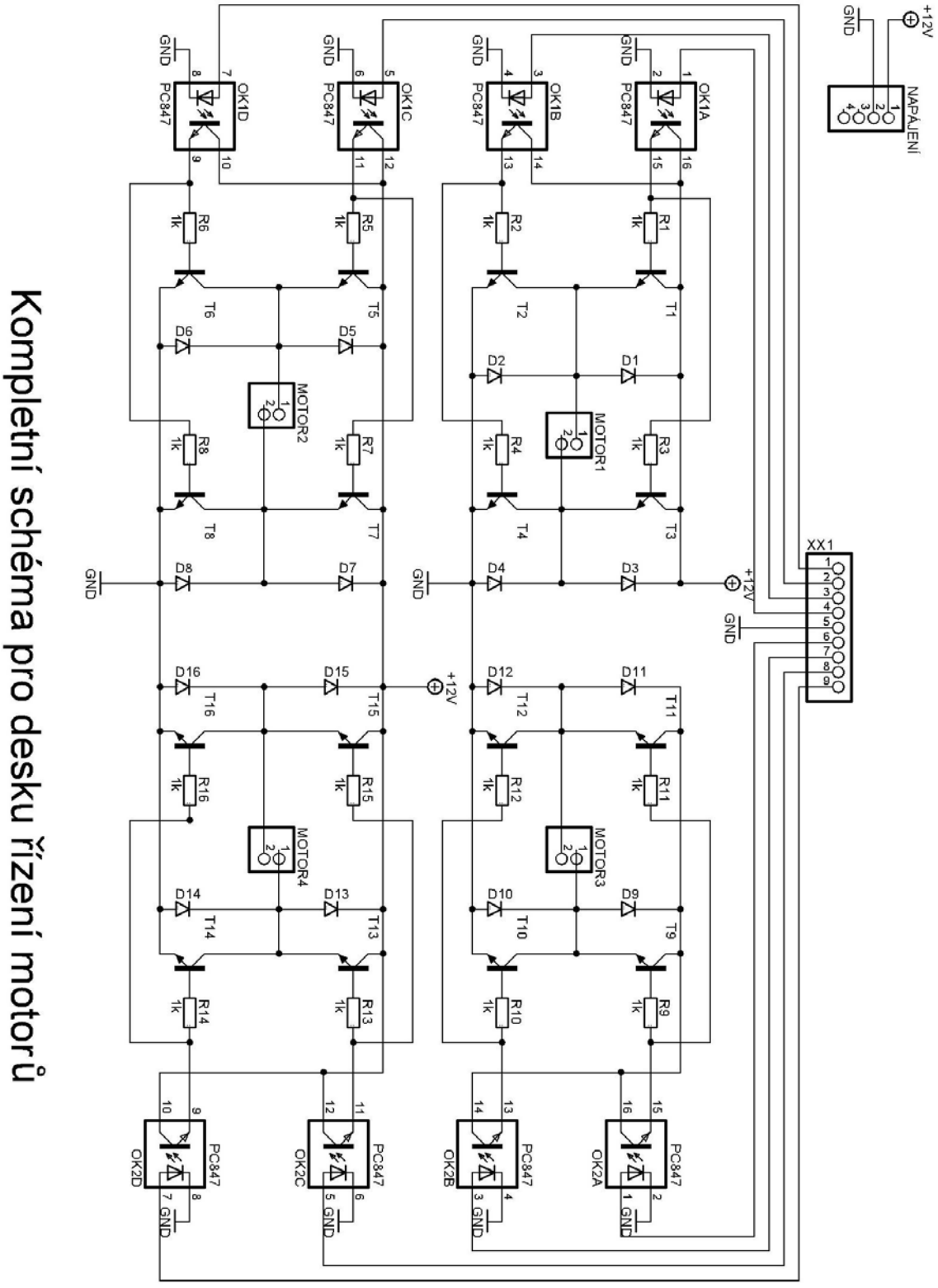
Tabluka P2.2 Popis konektorů – Hlavní deska

X1	Přípojení k Řídící desce motorů – Výstup
X2	Připojení k PC – sériová port
X3	Připojení k vozíku
X4	Vstup na PORTA
X5	Vstup na PORTB
X6	Výstup na PORTC
X7	Připojení konektoru X4 a X5
X8	Konektor určen pro „jumpery”
X9	Připojení k Propojovací desce
X10	Přípojení k Řídící desce motorů – Vstup
X11, X12	Náhradní výstupy z PIC16F84A

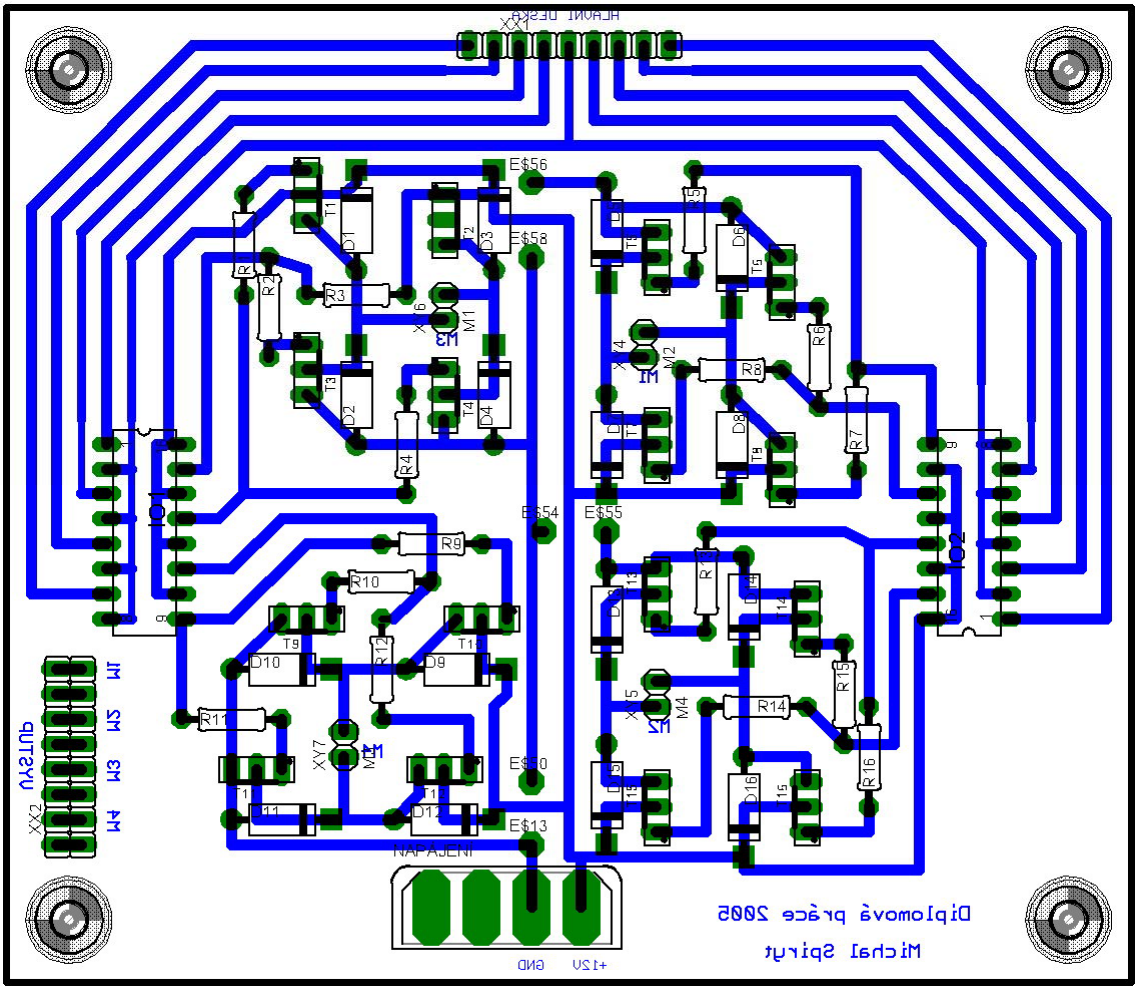


Obr.P2.3 Plošný spoj – Hlavní deska

Řídící deska motorů



Obr.P2.4 Úplné schéma zapojení desky pro řízení motorů



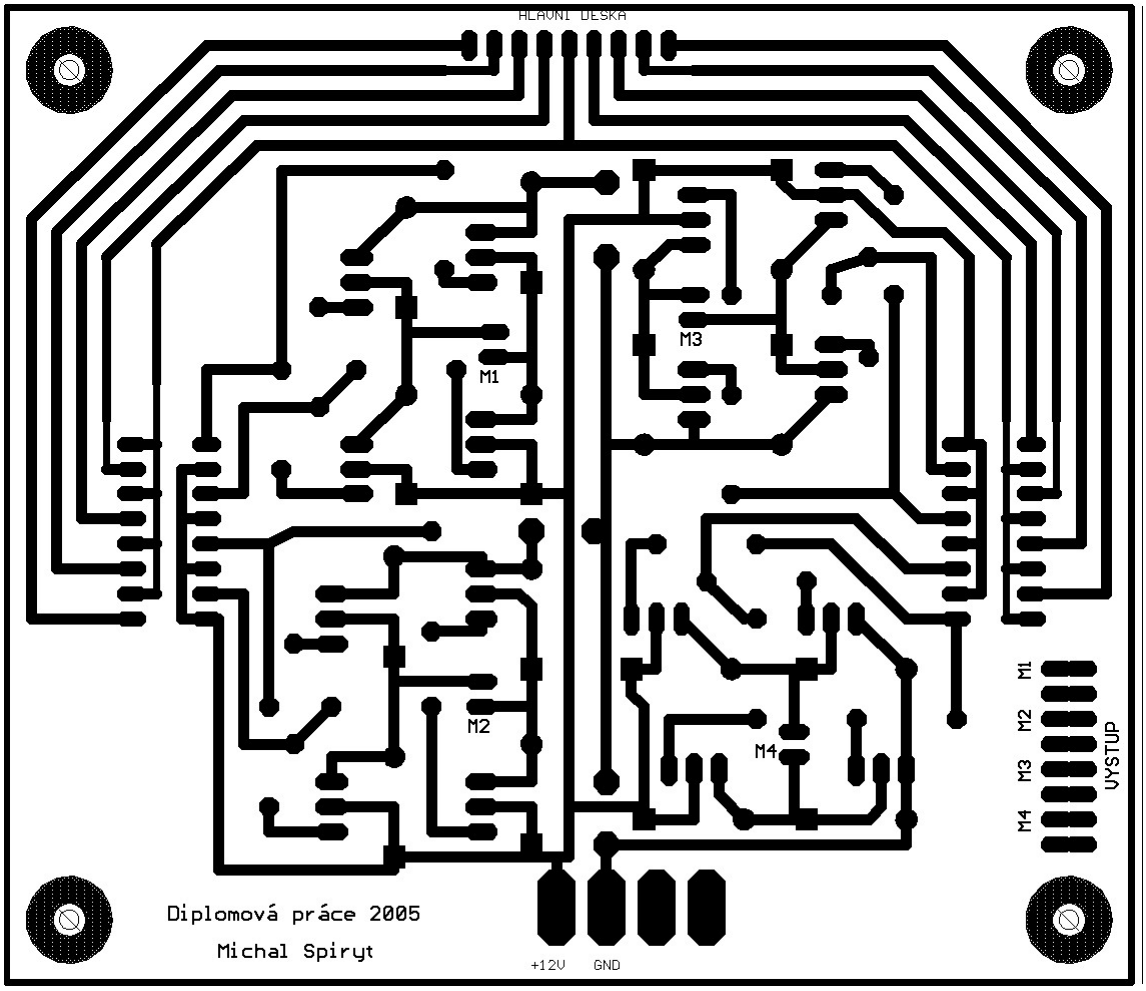
Obr.P2.5 Rozložení součástek na plošném spoji –Řídící deska motorů

Tabulka P2.3 Seznam součástek – Řídící deska motorů

R1 – R16	1k
T1 – T16	BD139-16
D1 – D16	1N4007
OK1, OK2	PC847

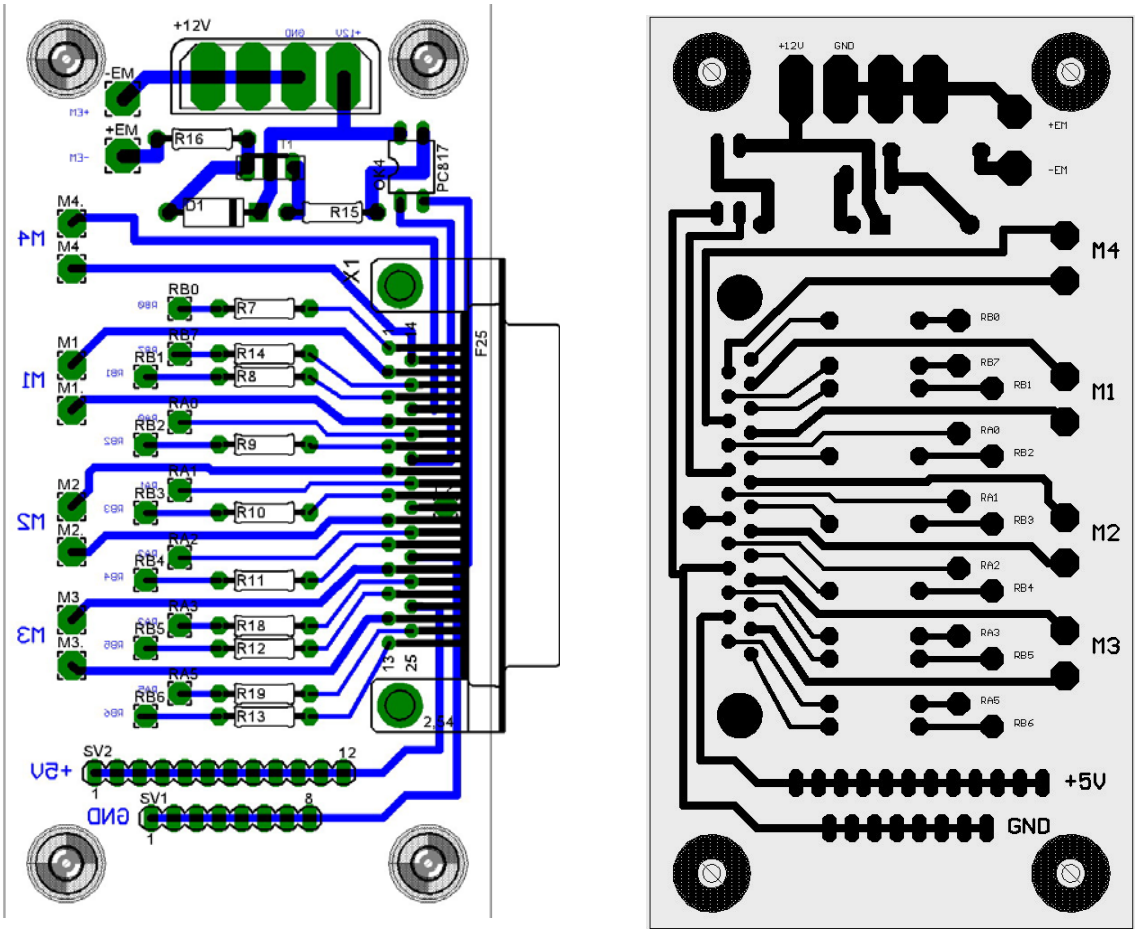
Tabulka P2.4 Popis konektorů - Řídící deska motorů

XX1	Připojení k Hlavní desce - Vstup
XX2	Připojení k Hlavní desce - Výstup



Obr.P2.6 Plošný spoj – Řídící deska motorů

Propojovací deska umístěná na jeřábu



a) b)

Obr.P2.7 a) Rozložení součástek na plošném spoji b) Plošný spoj - Propojovací deska

Tabulka P2.5 Seznam součástek – Propojovací deska

R7 – R14, R16, R18, R19	1k
R15	300R
D1	1N4007
T1	BD139-16
X1	CON 25

Zdrojový text souboru *diplom77.asm*

```
        LIST    P = 16F877, R = DEC
        #include<p16f877.inc>

;  ************    vlozeni souboru inc  *****************

        #include<zakmity.inc>
        #include<PosliNeboPrijmy.inc>
        #include<DolniPojezd.inc>
        #include<HorniPojezd.inc>
        #include<KomunikaceVozik.inc>

w       equ     0       ; instrukční výsledek do W
f       equ     1       ; instrukční výsledek do registru
rychlost    equ     D'64'

RAM             equ     020h            ; prvni adresa pameti RAM v PIC16F877(20h-7Fh)
TMP_W           equ     RAM+01          ; temp preruseni
TMP_S           equ     RAM+02          ; temp preruseni
TMP_PCL         equ     RAM+03          ; temp preruseni

TMP0            equ     RAM+04          ; pomocny registr
TMP1            equ     RAM+05          ; pomocny registr
TMP2            equ     RAM+06          ; pomocny registr
TMP3            equ     RAM+07          ; pomocny registr
TMP4            equ     RAM+08          ; pomocny registr


#define carry   STATUS,0 ; STATUS register
#define dcarry  STATUS,1
#define zero    STATUS,2

;************************ PROMENE ***************************************
;           !!! vynulovani a nastaveni promenych !!!
OTACKYL         equ     RAM+09          ; vzdalenost leveho motoru( deleno 8)
OTACKYP         equ     RAM+10          ; vzdalenost praveho motoru( deleno 8)
OTACKYH         equ     RAM+11          ; vzdalenost horniho motoru( deleno ?)
ELMAG           equ     RAM+12          ; zapnuty (1) nebo vypnuty (0) elektromagnet

OTACKYLPPC  equ     RAM+13          ; vzdalenost zadana z PC lev. a prav. motoru pro dojezd ( X )
OTACKYHPC   equ     RAM+14          ; vzdalenost zadana z PC horniho motoru pro dojezd na
1.misto ( Y )
ELMAGPC         equ     RAM+15          ; pozadavek z PC na zpnuty(1) nebo vypnuty(0)
elektromagnet po odviti
VOZIKPC         equ     RAM+16          ; pozice voziku zadana z PC

VOZIK           equ     RAM+17          ; aktualni pozice voziku

PRIR            equ     RAM+18          ; 0 bit indikuje rovnost OTACKYPC a OTACKY u leveho a
praveho motoru
                                        ; 1 bit indikuje rovnost OTACKYPC a OTACKY u horniho
motoru

PREDCHOZI       equ     RAM+19          ; 0 bit indikuje, co bylo pred tim na snimaci u leveho motorku
                                        ; 1 bit indikuje, co bylo pred tim na snimaci u praveho motorku
                                        ; 2 bit indikuje, co bylo pred tim na snimaci u horniho motorku
POCET           equ     RAM+20          ; pocet cyklu pro opakovani nacitani portu (kvuli prekmitu]
PREDPORT        equ     RAM+21          ; kopie portu..pouziti u kontroly prekmitu
```

```
; Seznam potrebnych VSTUPU A VYSTUPU >>>
;       zapnuti motorku:  RD3,RD4,RD6,RD7  1-jede,0-stoji
;               (  RD6, RD7 - levy a pravy motor, pokyny pro PIC16F84 )
;               (  RD3, RD4 - horni a elmag motor, pokyny pro PIC16F84 )
;       smer motorku: RD2, RD5  1-jede vlevo,0-jede vpravo
;       snimac poctu otoceni: SNIMACL,SNIMACP,SNIMACH
;               ( SNIMACL - otacky na levym motoru , SNIMACP - otacky na pravym motoru )
;               ( SNIMACH- otacky horniho motoru
;****************************** VYSTUPY *********************************

#define RD0     PORTD,0             ; vystup na vozik pin 3
#define RD1     PORTD,1             ; vystup na vozik pin 2
;-----------------------------------------------------------------
#define RD2     PORTD,2             ; vystup na RA1 pro horni a elmag. motor - urcuje smer
otaceni
#define RD3     PORTD,3             ; vystup na RA2 pro horni - urcuje zda pojede nebo ne
#define RD4     PORTD,4             ; vystup na RA3 pro elmag. - urcuje zda pojede nebo ne
#define RD5     PORTD,5             ; vystup na RA1 pro levy a pravy motor - urcuje smer otaceni
#define RD6     PORTD,6             ; vystup na RA2 pro levy - urcuje zda pojede nebo ne
#define RD7     PORTD,7             ; vystup na RA3 pro pravy - urcuje zda pojede nebo ne

;****************************** VSTUPY *********************************
#define RE0 PORTE,0             ; vstup z voziku pin 6
#define RE1 PORTE,1             ; vstup z voziku pin 5
#define RE2 PORTE,2             ; nahradni vstup
;-----------------------------------------------------------------
#define SNIMACL     PORTA,0     ;snimac leveho motorku  - vstup do RA0
#define SNIMACP     PORTA,1     ;snimac praveho motorku - vstup do RA1
#define SNIMACH     PORTA,2     ;snimac horniho motorku - vstup do RA2
;-----------------------------------------------------------------
#define ELMH    PORTB,1             ; indikator previti
#define  KONCLD     PORTB,2         ; doplnkovy snimac dolnich motorku na levo(bude na prave
kolejnici)
#define  KONCPD     PORTB,3         ; doplnkovy snimac dolnich motorku na pravo(bude na prave
kolejnici)
#define  KONCL PORTB,4              ; koncovy snimac dolnoch motorku na levo
#define  KONCP PORTB,5              ; koncovy snimac dolnoch motorku na pravo
#define  KONCHL     PORTB,6         ; koncovy snimac horniho motorku na levo
#define  KONCHP     PORTB,7         ; koncovy snimac horniho motorku na pravo
;-----------------------------------------------------------------
#define RC2             PORTC,2     ; timto se zapina(1) a vypina(0) elektromagnet

;*****************************************************************************

; Zakladni konfiguracni slovo
        org     0x2007          ; adresa konfigurace PIC16F877
        __CONFIG    B'11111100111010'



;******************** Reset procesoru ********************

RESET org       0x00            ; zacatek programu
        goto    INIT            ; skok na pocatecni inicializaci

        org     0x04            ; adresa preruseni
        goto    PRERUS
```

```
;********************      Přerusení      ********************

PRERUS          movwf   TMP_W           ; ulozi W
                movf    STATUS,w
                movwf   TMP_S           ; ulozi STATUS
;-----------------------------------------------------
                btfss   INTCON,RBIF
                goto    DAL1

                btfss   KONCL
                goto    $+2
                call    ODJETIDOLNILEVE

                btfss   KONCP
                goto    $+2
                call    ODJETIDOLNIPRAVE


                btfss   KONCHL
                goto    $+2
                call    ODJETIHORNILEVE

                btfss   KONCHP
                goto    $+2
                call    ODJETIHORNIPRAVE


                bcf             INTCON,RBIF
                goto    INT_END

DAL1    goto INT_END

;       !!! zde se napise co ma vykonat dalsi typ preruseni !!!


;-------------------------------------------------
INT_END movf  TMP_S,w
                movwf   STATUS                  ; obnova STATUS

                swapf   TMP_W,F                 ; prohodi nibble TMP_W a ulozi do W
                swapf   TMP_W,W                 ; obnova W
                retfie

;******************** Inicializace procesoru  ********************

INIT    bcf     STATUS,RP0      ; nastaveni BANKY 0
                bcf     STATUS,RP1      ; nastaveni BANKY 0
;       ------------------------
                movlw   b'000000'       ; prednastaveni PORTu A
                movwf   PORTA
                movlw   b'00000000'     ; prednastaveni PORTu B
                movwf   PORTB
                movlw   b'00000000'     ; prednastaveni PORTu C
                movwf   PORTC
                movlw   b'00000000'     ; přednastavení PORTu D
                movwf   PORTD
                movlw   b'000'          ; přednastavení PORTu E
                movwf   PORTE
;       ------------------------
                bsf     STATUS,RP0      ; nastaveni BANKY 1
```

```
                movlw   b'00000110'     ; A/D vstupy OFF
                movwf   ADCON1

                movlw   b'11010001'     ; PULL-UPy OFF, preddelic=1:4 pripojen k TMR0
                movwf   OPTION_REG      ; f:4  :4  :250 = 1000Hz (1ms)

                movlw   b'111111'       ; portA 5-0 5vstupy, 0vystupy
                movwf   TRISA

                movlw   b'11111111'     ; portB 7-0, 8vstupy, 0vystupy
                movwf   TRISB

                movlw   b'10000000'     ; portC 7-0, 1vstupy, 7vystupy
                movwf   TRISC

                movlw   b'00000000'     ; portD 7-0, 0vstupy, 8vystupy
                movwf   TRISD

                movlw   b'111'          ; portE 2-0, 3vstupy, 0vystupy
                movwf   TRISE

                bcf     STATUS,RP0      ; nastaveni BANKY 0
;       ------------------------
                clrf    TMR0            ; nulovat TMR0 (i jeho preddelic)

                movlw   b'10001000'     ; povolit preruseni pri zmene na vstupu RB4-RB7
                                        ; a pri jinych preruseni je vypnuty
                movwf   INTCON
;       ------------------------
;*****************************************************************


                goto MAIN

;#################### Casova smycka  ##############################################
;pouziva se vzdy pro dojezd, nez se zmeni smer otaceni

TIME    ;odpovida casu cca 1s
        MOVLW 0x50  ;80 DEC
        MOVWF TMP2
        MOVLW 0x7E  ;126 DEC
        MOVWF TMP1
        MOVLW 0x0A4  ;164 DEC
        MOVWF TMP0
        DECFSZ TMP0,F
        GOTO $-1
        DECFSZ TMP1,F
        GOTO $-5
        DECFSZ TMP2,F
        GOTO $-9
            return
;#################################################################################################

;##############  Procedury pro preruseni z RB4 az RB7  ############################
;***************  Casova smycka pro dojezd na nultou a maximalni polohu  ************
;
NULA    ; odpovida casu cca 0.5s
              MOVLW 0x28  ;40 DEC
              MOVWF TMP2
              MOVLW 0x7E  ;126 DEC
```

```
	MOVWF TMP1
	MOVLW 0x0A4  ;164 DEC
	MOVWF TMP0
	DECFSZ TMP0,F
	GOTO $-1
	DECFSZ TMP1,F
	GOTO $-5
	DECFSZ TMP2,F
	GOTO $-9
		return

;#########################################################################################

;############   Procedury pro navijeni a odvijeni navijaku   #################
;******** Cekajici smycka pro navijeni a odvijeni navijaku pro elektromagnet
DOBA   ; odpovida casu cca 5s
	MOVLW 0x0FF  ;255 DEC
	MOVWF TMP2
	MOVLW 0x0FF  ;255 DEC
	MOVWF TMP1
	MOVLW 0x0FE  ;254 DEC
	MOVWF TMP0
	DECFSZ TMP0,F
	GOTO $-1
	DECFSZ TMP1,F
	GOTO $-5
	DECFSZ TMP2,F
	GOTO $-9
	    return

;#########################################################################################
;*******  Navijeni a odvijeni  **********************************************
; navije a odvije lano s elektromagnetem, v dolni poloze bud zapne nebo
; vypne elektromagnet
NAVIJAODVIJ
		bcf	RD3		; pro jistotu vypina horni pojezd
		bsf	RD2
		bsf	RD4
		call	DOBA
		bcf	RD4
		btfsc	ELMAGPC,0
		bsf	RC2
		btfss	ELMAGPC,0
		bcf	RC2
		bcf	RD2
		call	TIME
		bsf	RD4
		call	DOBA
		bcf	RD4
		bcf	RD2
		call	TIME
		return
;#########################################################################################

	POJEZDDOLNI		;vklada vsechny procedury pro pohyb dolniho pojezdu
	POJEZDHORNI		;vklada vsechny procedury pro pohyb horniho pojezdu
```

MAIN

```
;********************    Hlavní program      ********************


                ;********  NAJETI DO NULOVE POLOHY PO STARTU  ********************
                        ; ****  DOLNI POJEZD  ****
                bcf     RD5
                bsf     RD6
                bsf     RD7
                movlw   .1
                movwf   OTACKYL
                btfsc   OTACKYL,0       ; dokud nedojde k preruseni, tak se OTACKYL nevynuluji
                goto    $-1                             ; z toho plyne smycka
                bcf     RD6
                bcf     RD7
                call    TIME
                        ; ****  HORNI POJEZD  ****
                bsf     RD2
                bsf     RD3
                movlw   .1
                movwf   OTACKYH
                btfsc   OTACKYH,0       ; dokud nedojde k preruseni, tak se OTACKYH nevynuluji
                goto    $-1
                bcf     RD3
                call    TIME
                        ; ***  NAVIJAK  **********
                bcf     RD2
                bsf     RD4
                btfss   ELMH
                goto    $-1
                bcf     RD4
                bcf     RC2
                call    TIME


        ;*************************************************************************

ZACATEK
                ; Tady je komunikace s PC,nejdriv se nastavi komunikace
                ; nejdriv mu nahlasi aktualni polohu
                ; a pote prijme pozadovanou
                INITKOM         rychlost

                ODESLI          OTACKYL         ; odesle hodnotu registru OTACKYL na PC
                ODESLI          OTACKYH         ; odesle hodnotu registru OTACKYH na PC
                ODESLI          ELMAG           ; odesle hodnotu registru ELMAG na PC
                ODESLI          PORTE           ; odesle hodnotu registru PORTE na PC

                PRIJMY          OTACKYLPPC  ; prijme znak ze serioveho portu a ulozi ho do
registru
                PRIJMY          OTACKYHPC   ; OTACKYLPPC, OTACKYHPC, ELMAGPC,
VOZIKPC
                PRIJMY          ELMAGPC
                PRIJMY          VOZIKPC

                NAVEDVOZIK VOZIKPC,PORTD        ; navedeni voziku do zadane polohy
```

```
;******** ZAHAJENI JIZDY DOLNIHO POJEZDU BUD VLEVO NEBO VPRAVO ******

        clrf    PRIR
        movf    OTACKYL,w
        subwf   OTACKYLPPC,w
        btfss   carry           ; zjistuje,jestli je pozadovana poloha vlevo nebo
        goto    $+4             ; vpravo od skutecne
        bsf     RD5
        call    JIZDAVLEVO
        goto    $+3
        bcf     RD5
        call    JIZDAVPRAVO
        bcf     RD6
        bcf     RD7
        call    TIME
        bcf     RD5

;*************************************************************************

;******* ZAHAJENI JIZDY HORNIHO POJEZDU BUD VLEVO NEBO VPRAVO ***********

        clrf    PRIR
        movf    OTACKYH,w
        subwf   OTACKYHPC,w
        btfss   carry           ; zjistuje,jestli je pozadovana poloha vlevo nebo
        goto    $+4             ; vpravo od skutecne
        bcf     RD2
        call    JIZDAVPRAVOHORNI
        goto    $+3
        bsf     RD2
        call    JIZDAVLEVOHORNI
        bcf     RD3
        call    TIME
        bcf     RD2

;*************************************************************************

        VOZIKOK     PORTE,VOZIKPC       ; testuje, zda-li je vozik na zadane poloze

;******  ZACATEK ODVIJENI A NASLEDNEHO NAVIJENI elmag motoru ************

        btfsc   RC2             ; kdyz je elmag zapnut,nastavi 1.bit ELAMGPC na 1
        bsf     ELMAGPC,1
        movlw   .0
        addwf   ELMAGPC,w
        btfsc   zero            ; kdyz je 1. i 0. bit nataven na 0,
        goto    $+6             ;skoci program az za odvijeni a navijeni
        movlw   .3
        subwf   ELMAGPC,w       ; kdyz je 1. i 0. bit nastaven na 1,
        btfsc   zero            ;skoci program az za odvijeni a navijeni
        goto    $+2
        call    NAVIJAODVIJ
        call    TIME

;*************************************************************************

        goto    ZACATEK

        end
```

Zdrojový text souboru *DolniPojezd.inc*

```
; zde jsou vsechny funkce urcene pro dolni pojezd

POJEZDDOLNI macro
;#########################################################################

;************** Pro preruseni od leve strany dolniho pojezdu + nasledne odjeti ******************

ODJETIDOLNILEVE
                bsf     RD5             ; zapne levy smer jizdy
                bcf     RD6             ; vypne pohon leveho motoru
                bsf     RD7             ; zapne pohon praveho motoru
                btfss   KONCLD          ; dokud se nesepne koncovy snimac na prave strane, stale
pravy motor jede
                goto    $-1
                bcf     RD7             ; vypne pohon praveho motoru
                call    TIME
                bcf     RD5             ; zapne pravy smer jizdy
                bsf     RD6     ; zapne pohon leveho motoru
                bsf     RD7     ; zapne pohon praveho motoru
                call    NULA
                bcf     RD6             ; vypne pohon leveho motoru
                bcf     RD7             ; vypne pohon praveho motoru
                call    TIME
                movlw   .100    ; !!!!! nutno spocitat - maximalni hodnota na dolnich kolejich
                movwf   OTACKYL
                movwf   OTACKYP
                return

;************** Pro preruseni od prave strany dolniho pojezdu + nasledne odjeti ******************

ODJETIDOLNIPRAVE
                bcf     RD5             ; zapne pravy smer jizdy
                bcf     RD6             ; vypne pohon leveho motoru
                bsf     RD7             ; zapne pohon praveho motoru
                btfss   KONCPD          ; dokud se nesepne koncovy snimac na prave strane, stale
                                        ;pravy motor jede
                goto    $-1
                bcf     RD7             ; vypne pohon praveho motoru
                call    TIME
                bsf     RD5             ; zapne levy smer jizdy
                bsf     RD6     ; zapne pohon leveho motoru
                bsf     RD7     ; zapne pohon praveho motoru
                call    NULA
                bcf     RD6             ; vypne pohon leveho motoru
                bcf     RD7             ; vypne pohon praveho motoru
                call    TIME
                clrf    OTACKYL ; vynuluje otacky leveho dolniho motoru
                clrf    OTACKYP ; vynuluje otacky praveho dolniho motoru
                return

;************** PRO POHYB VPRED (VLEVO) ************************************
;************  Porovnava OTACKYL a OTACKYP a OTACKYLPPC  a nastoluje napravu *******

JIZDAVLEVO
                ;---------  Pro nacitani pulsu z leveho motoru pro jizdu vpred(vlevo)
                local   ZAC
                bsf     RD6
                bsf     RD7
```

```
ZAC             KMIT    PORTA,POCET,PREDPORT
                btfss   PREDPORT,0
                bcf     PREDCHOZI,0
                btfss   PREDPORT,0
                goto    $+4
                btfss   PREDCHOZI,0
                incf    OTACKYL,f
                bsf     PREDCHOZI,0
                ;-------- Pro nacitani pulsu z praveho motoru pro jizdu vpred(vlevo)
                btfss   PREDPORT,1
                bcf     PREDCHOZI,1
                btfss   PREDPORT,1
                goto    $+4
                btfss   REDCHOZI,1
                incf    OTACKYP,f
                bsf     PREDCHOZI,1

                call    DOLNIPOJEZDL
                btfss   PRIR,0;tohle indikuje, ze uz muzu prestat jet
                goto    ZAC
                return

DOLNIPOJEZDL
                movf    OTACKYL,w
                subwf   OTACKYP,w
                btfsc   zero
                goto    POROVNANIPCL
                btfsc   carry
                goto    $+3
                bcf     RD6
                goto    DOLNIPOJEZDLK
                bcf     RD7
                goto    DOLNIPOJEZDLK
POROVNANIPCL
                bsf     RD6
                bsf     RD7
                movf    OTACKYLPPC,w
                subwf   OTACKYL,w
                btfsc   zero
                bsf     PRIR,0
DOLNIPOJEZDLK
                return

;************** PRO POHYB VZAD (VPRAVO) ************************************

JIZDAVPRAVO
                ;---------  Pro nacitani pulsu z leveho motoru pro jizdu vpred(vlevo)
                bsf     RD6
                bsf     RD7
ZAC1    KMIT    PORTA,POCET,PREDPORT
                btfss   PREDPORT,0
                bcf     PREDCHOZI,0
                btfss   PREDPORT,0
                goto    $+4
                btfss   PREDCHOZI,0
                decf    OTACKYL,f
                bsf     PREDCHOZI,0
                ;-------- Pro nacitani pulsu z praveho motoru pro jizdu vpred(vlevo)
                btfss   PREDPORT,1
                bcf     PREDCHOZI,1
```

```
                btfss   PREDPORT,1
                goto    $+4
                btfss   PREDCHOZI,1
                decf    OTACKYP,f
                bsf     PREDCHOZI,1

                call    DOLNIPOJEZDP
                btfss   PRIR,0;tohle indikuje, ze uz muzu prestat jet
                goto    ZAC1
                return

DOLNIPOJEZDP
                movf    OTACKYL,w
                subwf   OTACKYP,w
                btfsc   zero
                goto    POROVNANIPCP
                btfsc   carry
                goto    $+3
                bcf     RD7
                goto    DOLNIPOJEZDPK
                bcf     RD6
                goto    DOLNIPOJEZDPK
POROVNANIPCP
                bsf             RD6
                bsf             RD7
                movf    OTACKYL,w
                subwf   OTACKYLPPC,w
                btfsc   zero
                bsf             PRIR,0
DOLNIPOJEZDPK
                return

;###############################################################################

                endm
```

Zdrojový text souboru *HorniPojezd.inc*

```
; zde jsou vsechny funkce potrebne pro horni pojezd

POJEZDHORNI macro
;###############################################################################

;************* Pro preruseni od leve strany horniho pojezdu + nasledne odjeti ************

ODJETIHORNILEVE
                bcf     RD4             ; pro jistotu vypina navijeni
                bcf     RD3             ; vypne horni motor
                call    TIME
                bcf     RD2             ; zapne pravy smer horniho motoru
                bsf     RD3             ; zapne horni motor
                call    NULA
                bcf     RD3             ; vypne horni motor
                call    TIME
                clrf    OTACKYH         ; vynuluje OTACKY horniho motoru
                return

;************* Pro preruseni od prave strany horniho pojezdu + nasledne odjeti ************

ODJETIHORNIPRAVE
                bcf     RD4             ; pro jistotu vypina navijeni
                bcf     RD3             ; vypne  horni motor
                call    TIME
                bsf     RD2             ; zapne levy smer horniho motoru
                bsf     RD3             ; zapne horni motor
                call    NULA
                bcf     RD3             ; vypne horni motor
                call    TIME
                movlw   .35             ; maximalni horni otacky
                movwf   OTACKYH
                return

;*************  Jizda vlevo  ***********************************************

JIZDAVLEVOHORNI
                ;---------  Pro nacitani pulsu z horniho motoru pro jizdu vzad(vlevo)
                bsf     RD3
                KMIT    PORTA,POCET,PREDPORT
                btfss   PREDPORT,2
                bcf     PREDCHOZI,2
                btfss   PREDPORT,2
                goto    $+4
                btfss   PREDCHOZI,2
                decf    OTACKYH,f
                bsf     PREDCHOZI,2

                call    HORNIPOJEZDL
                btfss   PRIR,1;tohle indikuje, ze uz muzu prestat jet
                goto    $-10
                return



HORNIPOJEZDL
```

```
                movf    OTACKYHPC,w
                subwf   OTACKYH,w
                btfsc   zero
                bsf     PRIR,1
                return

;************  Jizda vpravo  ***********************************************

JIZDAVPRAVOHORNI
                ;---------  Pro nacitani pulsu z horniho motoru pro jizdu vpred(vpravo)
                bsf     RD3
                KMIT    PORTA,POCET,PREDPORT
                btfss   PREDPORT,2
                bcf     PREDCHOZI,2
                btfss   PREDPORT,2
                goto    $+4
                btfss   PREDCHOZI,2
                incf    OTACKYH,f
                bsf     PREDCHOZI,2

                call    HORNIPOJEZDP
                btfss   PRIR,1;tohle indikuje, ze uz muzu prestat jet
                goto    $-10
                return

HORNIPOJEZDP
                movf    OTACKYH,w
                subwf   OTACKYHPC,w
                btfsc   zero
                bsf     PRIR,1
                return

;###############################################################################

        endm
```

Zdrojový text souboru *KomunikaceVozik.inc*

```
; urcene pro komunikaci s vozikem

;******** NAVEDENI VOZIKU NA URCENOU POLOHU **************************

NAVEDVOZIK  macro   VOZIKPCX,PORTX

            movlw   .0              ; v pripade, ze je VOZIKPC 0 nastavuje RD1 a RD0
            addwf   VOZIKPCX,w  ; nastavuje depo
            btfss   zero
            goto    $+3
            bsf             PORTX,1         ; nastavuju, protoze vozik to vnima naopat (ST)
            bsf             PORTX,0
            movlw   .1              ; v pripade, ze je VOZIKPC 1 nastavuje RD1 a nuluje RD0
            subwf   VOZIKPCX,w  ; nastavuje 1.nadrazi
            btfss   zero
            goto    $+3
            bsf             PORTX,1
            bcf             PORTX,0
            btfss   carry   ; v pripade, ze je VOZIKPC 2 nuluje RD1 a nastavuje RD0
            goto    $+3                     ; natavuje 2.nadrazi
            bcf             PORTX,1
            bsf             PORTX,0 ; nastaveni trva az do skonceni vsech pojezdu

            endm

;***********************************************************************

;********* JE VOZIK NA URCENEM MISTE? *********************************

VOZIKOK     macro   PORTX,VOZIKPCX

            movf    PORTX,w
            subwf   VOZIKPCX,w
            btfss   zero            ; dokude neni vozik na zadanem miste, ceka se !
            goto    $-3

            endm

;***********************************************************************
```

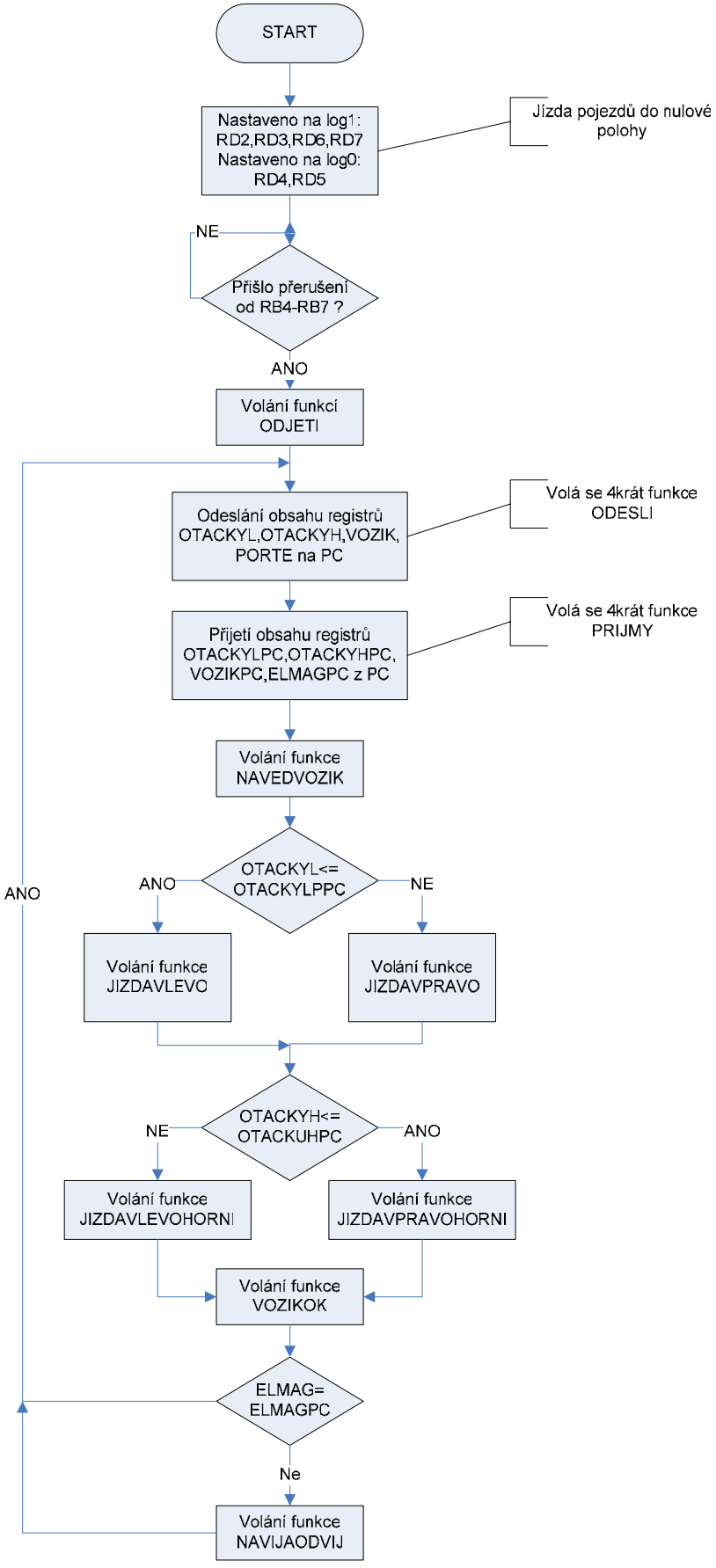
Zdrojový text souboru *PosliNeboPrijmy.inc*

```
; ***************  Inicializuje prenos po USART  *********************************

INITKOM         macro   rych
                        ; nastaveni registru          TXSTA, RCSTA, SPBRG> 8 bitovy prenos
                        ; (vysilac i prijimac) s rychlosti 19200 baudu
                movlw   B'00100100'     ; nastav vysilac pro...
                banksel TXSTA           ; ...asynchronni 8bitovy prenos
                movwf   TXSTA
                movlw   B'10010000'     ; nastav prijimac pro...
                banksel RCSTA           ; ...asynchronni 8bitovy prijem
                movwf   RCSTA
                movlw   rych            ; nastav komunikacni rychlost,...
                banksel SPBRG           ; ...v nasem pripade 19200 baudu
                movwf   SPBRG
                banksel 0

                endm

;*******************************************************************************

;*************  Toto makro odesila znak na PC  *********************************

ODESLI          macro   CISLO

                btfss   PIR1,TXIF       ; pokud je vysilaci port prazdny
                goto    $-1
                movf    CISLO,w                 ;...vysli CISLO
                movwf   TXREG

                endm

;*******************************************************************************

;**********************  Prijima znak z PC  ************************************

PRIJMY          macro   CISLO

                btfss   PIR1,RCIF               ; pokud na port prisel znak...
                goto    $-1
                movf    RCREG,W                 ; ...uloz jej do registru CISLO
                movwf   CISLO

                endm

;*******************************************************************************
```

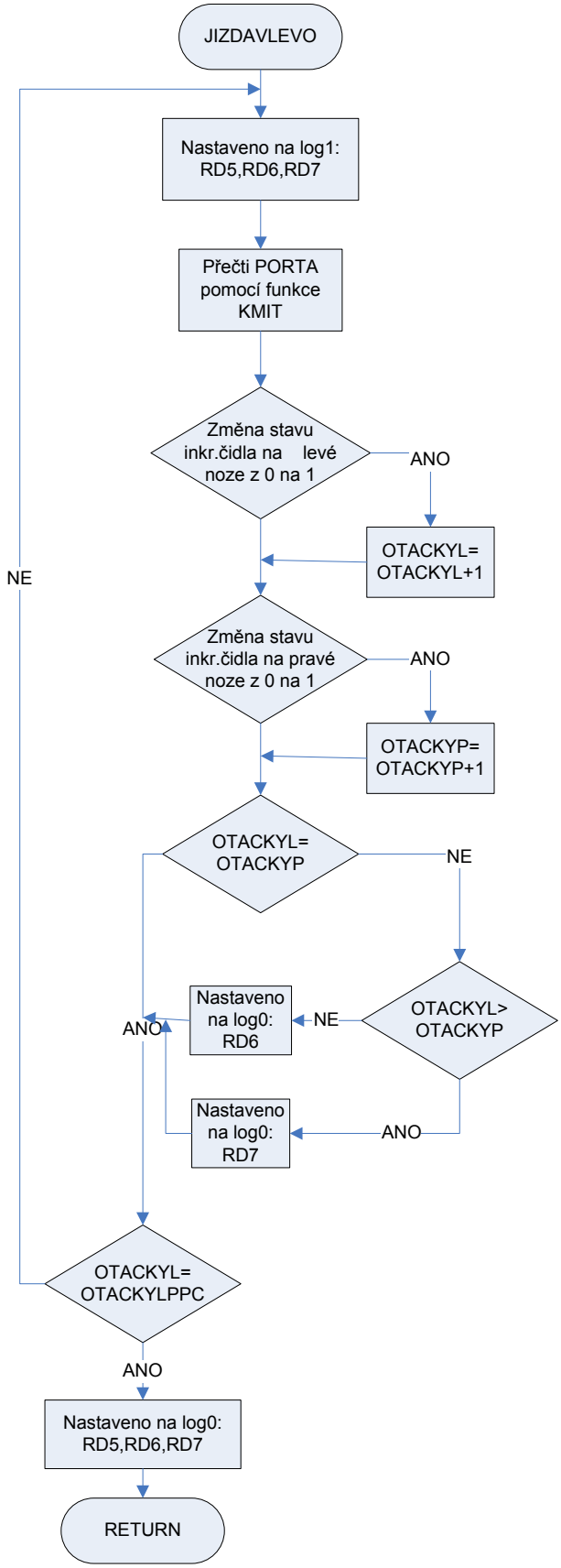
Zdrojový text souboru *zakmity.inc*

```
; Toto makro slouzi k osetreni zakmitu vstpu.
; Cteni urceeho portu opakuje 20 krat a v pripade, ze je stale stejny,
; vezme jeho hodnotu a urcujici.
;*****************************************************************
; PORTX ....vlozit port, ktery je nutno testovat
; POCETX.... 8 bitovy register, ktery je urceny pro odcitani poctu opakovani
; UCHOVAT....8 bitovy registr, do ktereho j epotreba ulozit urcujici hodnotu portu

KMIT macro      PORTX,POCETX,UCHOVAT

                movlw   20
                movwf   POCETX
                movf    PORTX,w
                movwf   UCHOVAT

                movf    PORTX,w
                subwf   UCHOVAT,w
                btfss   zero
                goto    $-7
                decfsz  POCETX,f
                goto    $-5

        endm
```

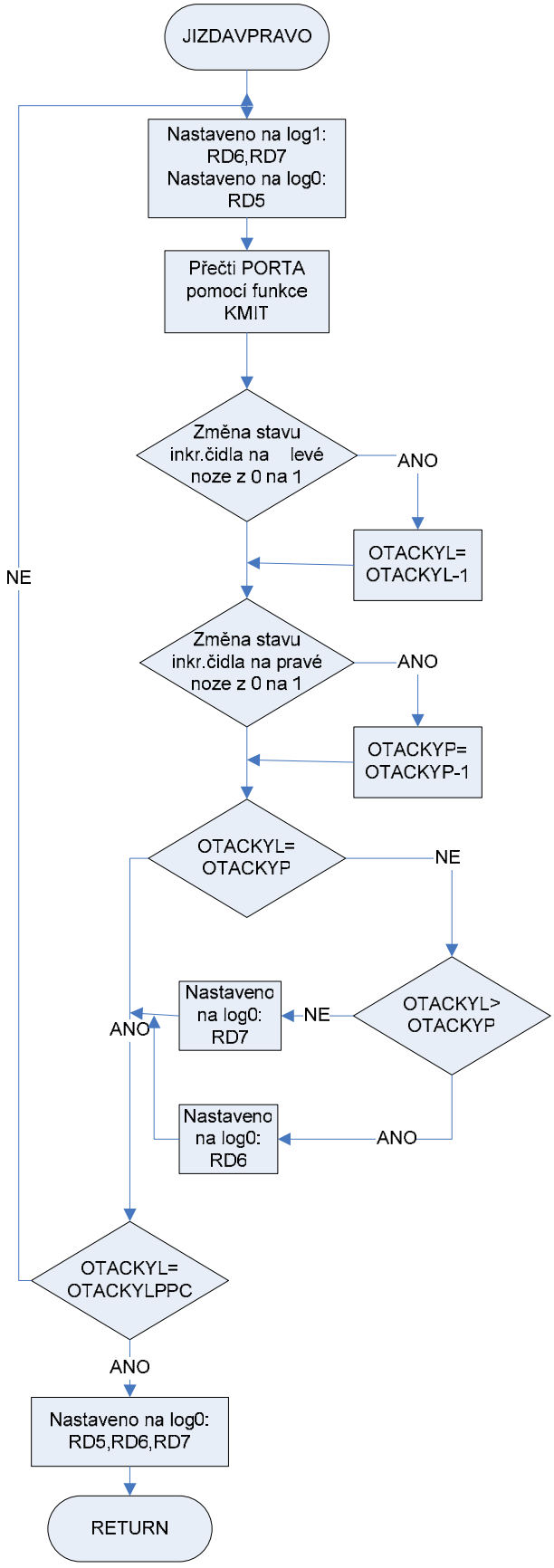
Vývojový diagram chování řídícího mikropočítače



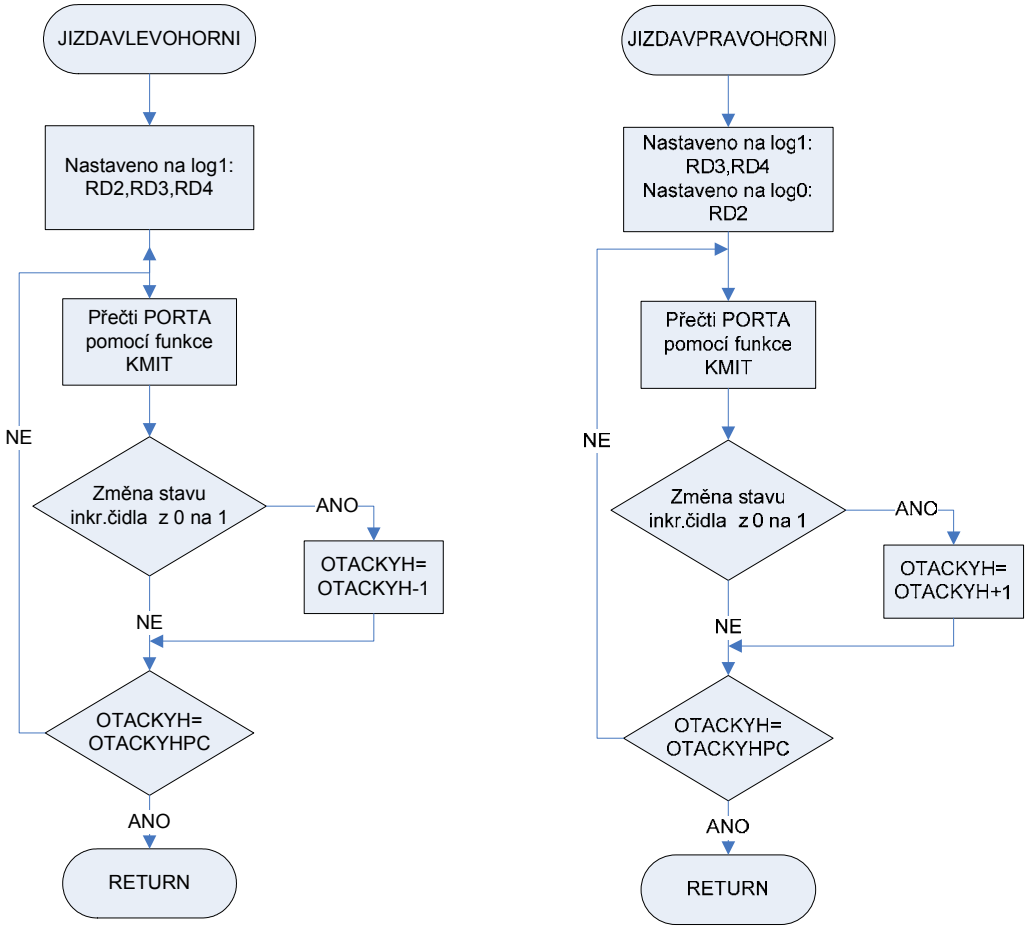
Vývojový diagram funkce JIZDAVLEVO



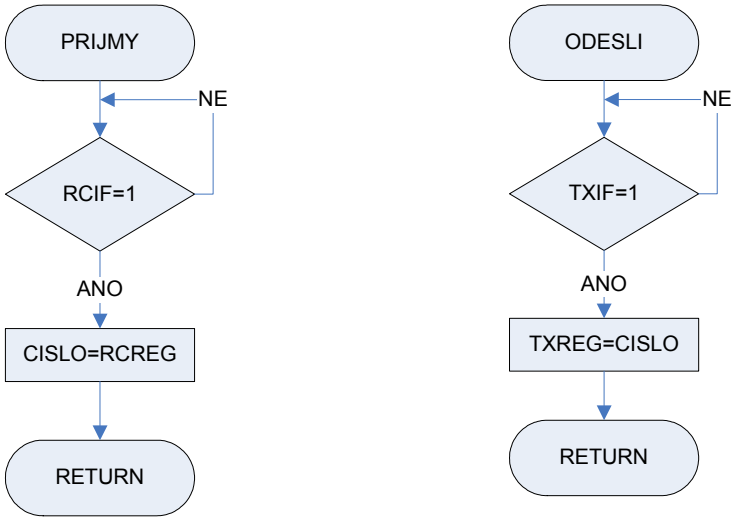
Vývojový diagram funkce JIZDAVPRAVO



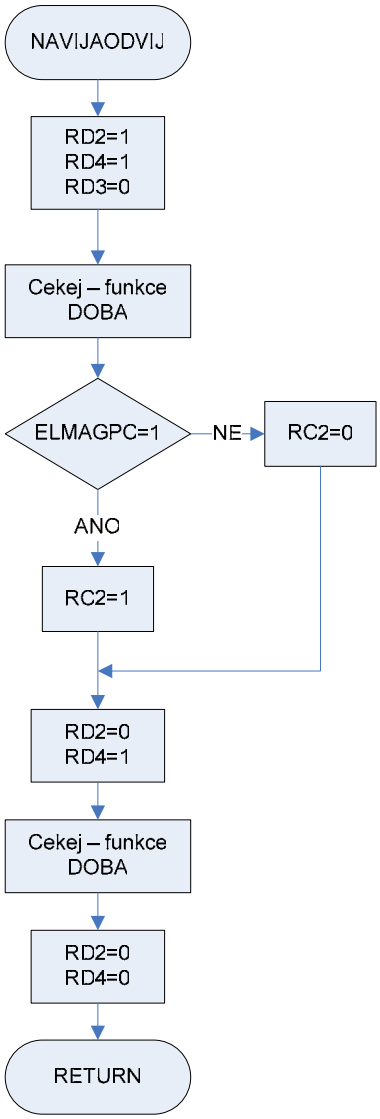
Vývojový diagram funkce JIZDAVLEVOHORNI a JIZDAVPRAVOHORNI



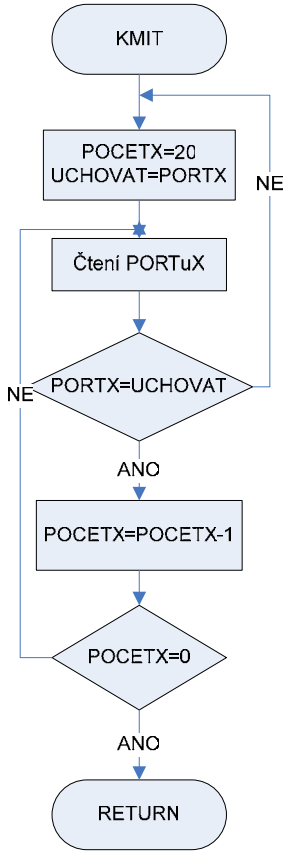
Vývojový diagram funkce PRIJMY a ODESLI



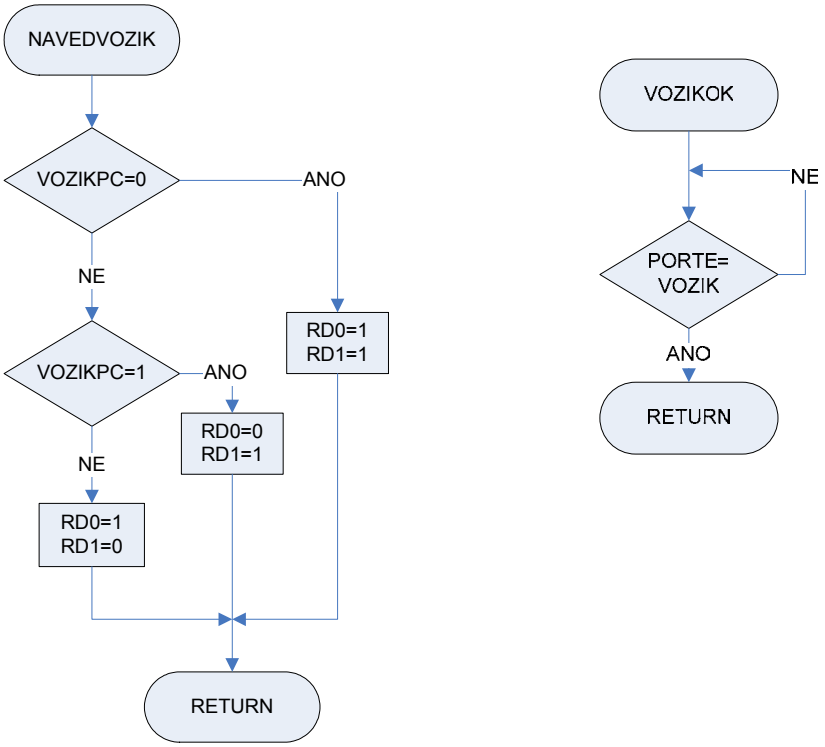
Vývojový diagram funkce NAVIJAODVIJ



Vývojový diagram funkce KMIT



Vývojový diagram funkce NAVEDVOZIK a VOZIKOK



Zdrojový text souborů *diplom84.asm*

```
        LIST    P = 16F84, R = DEC
        #include<p16f84.inc>


w       equ     0       ; instrukční výsledek do W
f       equ     1       ; instrukční výsledek do registru

RAM             equ     0Ch             ; prvni adresa RAM v PIC16F84 (0Ch-2Fh)
TMP_W           equ     RAM+01          ; temp preruseni
TMP_S           equ     RAM+02          ; temp preruseni

TMP0            equ     RAM+03          ; pomocny registr
TMP1            equ     RAM+04          ; pomocny registr
TMP2            equ     RAM+05          ; pomocny registr

#define carry   STATUS,0 ; STATUS register
#define dcarry  STATUS,1
#define zero    STATUS,2

; *********   zakladni nastaveni mikrokontroleru  **********************

                org 0x2007      ; adresa konfigurace PIC16F84
                __config   0x3FF9

;*********************    Reset procesoru     *********************

RESET   org     0x00            ; zacatek programu
        goto    INIT            ; skok na pocatecni inicializaci

        org     0x04            ; adresa preruseni
        goto    PRERUS

;*********************       Přerusení      *********************

PRERUS          movwf   TMP_W           ; ulozit W
                movf    STATUS,W
                movwf   TMP_S           ; ulozit STATUS
;       ------------------------
                bcf     INTCON,T0IF     ; vynulovat priznak
;-------------------------------

;       !!! zde lze napsat,co ma udelat preruseni !!!

;-------------------------------
INT_END         movf    TMP_S,W
                movwf   STATUS                  ; obnova STATUS
                swapf   TMP_W,F
                swapf   TMP_W,W                 ; obnova W
                retfie                          ; navrat z preruseni

;********************* Inicializace procesoru  *********************

INIT    bcf     STATUS,RP0              ; nastaveni BANKY 0
                bcf     STATUS,RP1      ; nastaveni BANKY 0
;       ------------------------
                movlw   b'00000'        ; prednastaveni PORTu A
                movwf   PORTA
```

```
                movlw   b'00000000'     ; prednastaveni PORTu B
                movwf   PORTB
;       ------------------------
                bsf     STATUS,RP0      ; nastaveni BANKY 1
                movlw   b'11111'        ; portA 4-0, 5vstupu, 0vystupu
                movwf   TRISA
                movlw   b'00000000'     ; portB 7-0, 0vstupu, 8vystupu
                movwf   TRISB

                movlw   B'11011000'
                movwf   OPTION_REG      ; f:4  :16  :256 = 100Hz (preruseni = 10ms)
                bcf     STATUS,RP0      ; nastaveni BANKY 0
;       ------------------------
                movlw   b'00000000'     ; zakazani jakehokoliv preruseni
                movwf   INTCON

;*****************************************************************
;               !!! vynulovani a nastaveni PROMenych !!!


POMRA           equ     RAM+06          ; pomocny registr pro RA
POMRB           equ     RAM+07          ; pomocny registr pro RB

LEVY5           equ     RAM+08          ; pocet 5V ve stride prvni motor
PRAVY5          equ     RAM+09          ; pocet 5V ve stride druhy motor

POMLEVY5        equ     RAM+11          ; odecitana strida 5V prvni motor
POMPRAVY5       equ     RAM+12          ; odecitana strida 5V druhy motor

ULOZLEVY5       equ     RAM+13          ; ulozena LEVY5
ULOZPRAVY5      equ     RAM+14          ; ulozena PRAVY5

PROM            equ     RAM+15          ; pomocna PROMena
HLIDEJ          equ     RAM+16          ; pomocna PROMena hlida stalost vyvodu

PRIR            equ     RAM+17          ; detekuje, ze bylo prirazino LEVY5 do POMLEVY5 bit 0
                                        ; detekuje,ze bylo prirazino PRAVY5 do POMPRAVY5 bit 1
SECL            equ     RAM+18          ; pro zpomaleni plus a minus pro levy motor
SECP            equ     RAM+19          ; pro zpomaleni plus a minus pro pravy motor


#define         RA1     POMRA,3         ; vstup z RD5 PIC16F877
#define         RA2     POMRA,2         ; vstup z RD6 PIC16F877
#define         RA3     POMRA,1         ; vstup z RD7 PIC16F877

#define         RB4     PORTB,4         ; vystup na MOTORL vlevo
#define         RB5     PORTB,5         ; vystup na MOTORL vpravo
#define         RB6     PORTB,6         ; vystup na MOTORP vlevo
#define         RB7     PORTB,7         ; vystup na MOTORP vpravo

;********************    Hlavní program     ********************

                clrf    LEVY5           ; nulovani pracovnich registru
                clrf    PRAVY5
                clrf    POMLEVY5
                clrf    POMPRAVY5
                clrf    ULOZLEVY5
                clrf    ULOZPRAVY5
                clrf    PROM
                clrf    HLIDEJ
```

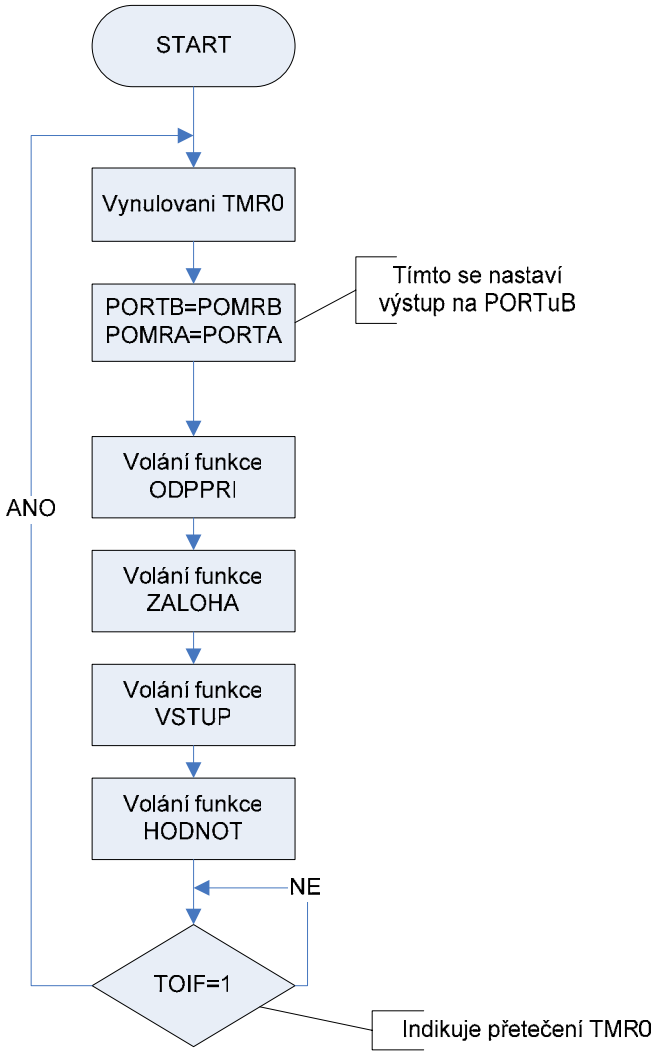
```
                clrf    POMRB
                clrf    PRIR
                movlw   .20
                movwf   SECL
                movwf   SECP

;*****************************************************************

                goto MAIN

;**********  ODCITANI a SCITANI plusovych casti stridy ******************

PLUS    incf    PROM,f          ; pricitani do registru PROM, max. 40
                movlw   .41
                subwf   PROM,w
                btfsc   carry
                decf    PROM,f
                return

MINUS   decf    PROM,f          ; odecitani z registru PROM, min. 0
                movlw   0xFF
                subwf   PROM,w
                btfsc   carry
                incf    PROM,f
                return

;***********    Kontrola nastaveni RA2,RA3 a nasladne vyhodnoceni ***********

VSTUP
                btfss   PRIR,0          ; testuje, zda-li je vstupni vyvod RA2 nastaven na 1 nebo 0
                goto    VSTUPP          ; 0 – vola MINUS, 1 –vola PLUS
                movf    LEVY5,w
                movwf   PROM
                btfss   RA2
                call    MINUS
                btfsc   RA2
                call    PLUS
                movf    PROM,w
                movwf   LEVY5

VSTUPP          btfss   PRIR,1          ; testuje, zda-li je vstupni vyvod RA3 nastaven na 1 nebo 0
                goto    VSTUPK          ; 0 – vola MINUS, 1 –vola PLUS
                movf    PRAVY5,w
                movwf   PROM
                btfss   RA3
                call    MINUS
                btfsc   RA3
                call    PLUS
                movf    PROM,w
                movwf   PRAVY5

VSTUPK          return

;***********  Zalohovani poctu kladnych casti stridy **********************

ZALOHA  movf            LEVY5,w                 ; uchovava sktualni stav registru LEVY a PRAVY
        movwf           ULOZLEVY5
        movf            PRAVY5,w
        movwf           ULOZPRAVY5
        return
```

```
;********** Pomoc pri rozjezdu motoru **********************************

ROZJEDL movlw   .4
        subwf   ULOZLEVY5,w
        btfss   zero
        goto    ROZLK
        movlw   .20
        movwf   POMLEVY5
ROZLK   return

ROZJEDP movlw   .4
        subwf   ULOZPRAVY5,w
        btfss   zero
        goto    ROZPK
        movlw   .20
        movwf   POMPRAVY5
ROZPK   return

;****************** Hodnoceni stridy a nasledne nastaveni POMRB ************

HODNOT  clrf    POMRB
        btfss   RA1
        goto    VPRAVO

; odtud hodnoti otaceni vlevo
        bcf     POMRB,4
        movlw   .251
        addwf   POMLEVY5,w
        btfsc   carry
        bsf     POMRB,4
        movlw   .40             ; odtud to hodnoti,zdalito ma zapnout furt +5V
        subwf   LEVY5,w
        btfsc   zero
        bsf     POMRB,4

        bcf     POMRB,6
        movlw   .251
        addwf   POMPRAVY5,w
        btfsc   carry
        bsf     POMRB,6
        movlw   .40             ; odtud to hodnoti,zdalito ma zapnout furt +5V
        subwf   PRAVY5,w
        btfsc   zero
        bsf     POMRB,6

        goto    HODNOTK

; odtud hodnoti otaceni vpravo

VPRAVO  bcf     POMRB,5
        movlw   .251
        addwf   POMLEVY5,w
        btfsc   carry
        bsf     POMRB,5
        movlw   .40             ; odtud to hodnoti,zdalito ma zapnout furt +5V
        subwf   LEVY5,w
        btfsc   zero
        bsf     POMRB,5

        bcf     POMRB,7
```

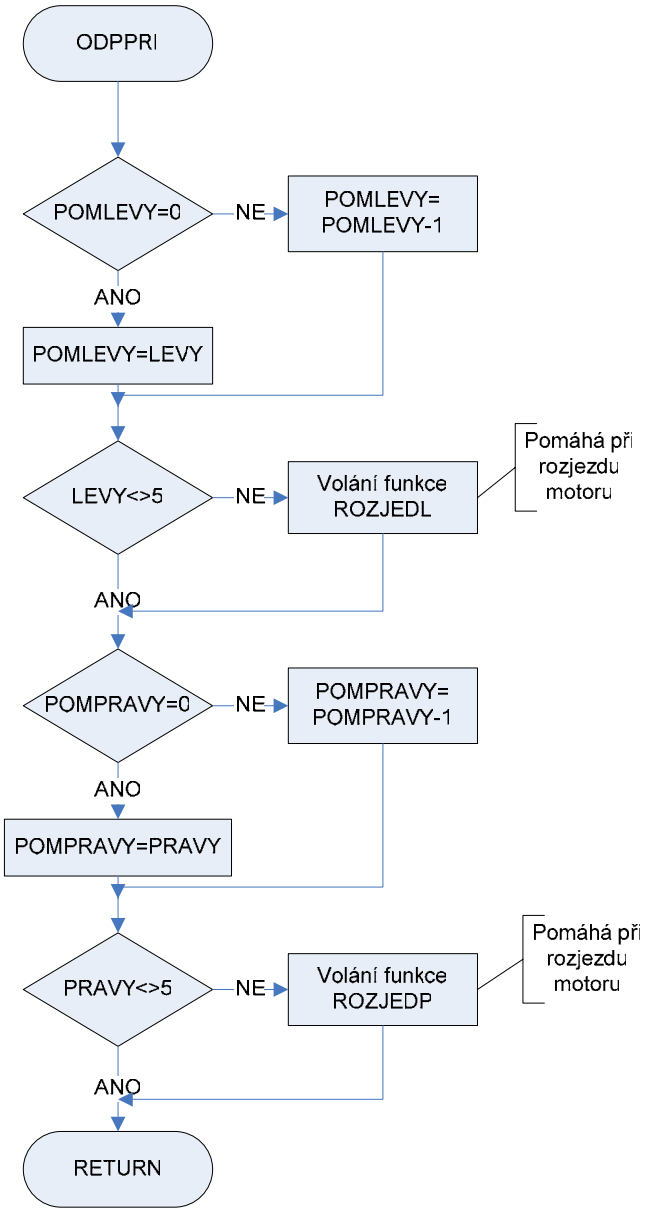
```
                movlw   .251
                addwf   POMPRAVY5,w
                btfsc   carry
                bsf     POMRB,7
                movlw   .40                 ; odtud to hodnoti,zdalito ma zapnout furt +5V
                subwf   PRAVY5,w
                btfsc   zero
                bsf     POMRB,7

HODNOTK         return

;*** Odescita promene POMLEVY(POMPRAVY) a priradi do nich LEVY(PRAVY) kdyz jsou =0   ***

ODPPRI          movlw   0x00
                addwf   POMLEVY5,w
                btfss   zero
                decf    POMLEVY5,f
                movf    STATUS,w
                movwf   HLIDEJ
                movf    LEVY5,w
                btfsc   HLIDEJ,2
                movwf   POMLEVY5
                btfss   HLIDEJ,2
                goto    $+2
                decfsz  SECL,f
                goto    $+4
                bsf     PRIR,0
                movlw   .20
                movwf   SECL
                movlw   .5
                subwf   LEVY5,w
                btfsc   zero
                call    ROZJEDL

                movlw   0x00
                addwf   POMPRAVY5,w
                btfss   zero
                decf    POMPRAVY5,f
                movf    STATUS,w
                movwf   HLIDEJ
                movf    PRAVY5,w
                btfsc   HLIDEJ,2
                movwf   POMPRAVY5
                btfss   HLIDEJ,2
                goto    $+2
                decfsz  SECP,f
                goto    $+4
                bsf     PRIR,1
                movlw   .20
                movwf   SECP
                movlw   .5
                subwf   PRAVY5,w
                btfsc   zero
                call    ROZJEDP

                return
```

```
;************************  Hlavni program **********************************

MAIN
        bcf     INTCON,2        ;vynulovani bitu T0IF
        clrf    TMR0            ;vynulovani TMR0 a preddelicky

        movf    POMRB,w         ; nastaveni PORTuB podle obsahu registru POMRB
        movwf   PORTB

        movf    PORTA,w         ; ulozeni hodnoty PORTuA do pomocneho registru POMRA
        movwf   POMRA


        call    ODPPRI
        call    ZALOHA
        call    VSTUP
        call    HODNOT

        bcf     PRIR,0
        bcf     PRIR,1
        movlw   .130
        addwf   TMR0,f
        btfss   INTCON,2        ;testovani bitu T0IF
        goto    $-1             ; dokud nepretece TMRO, jet u cekaci smycka, tim je zajistena stale
                                ; stajna delka prosleho programu MAIN a tim je take zajistena
                                ; presnost pulzni sirkove modulace
        goto    MAIN

        end
```

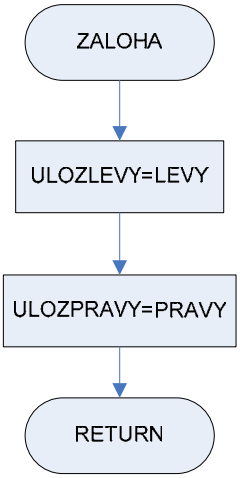
Vývojový diagram chování pomocných mikrokontrolérů PIC16F84A



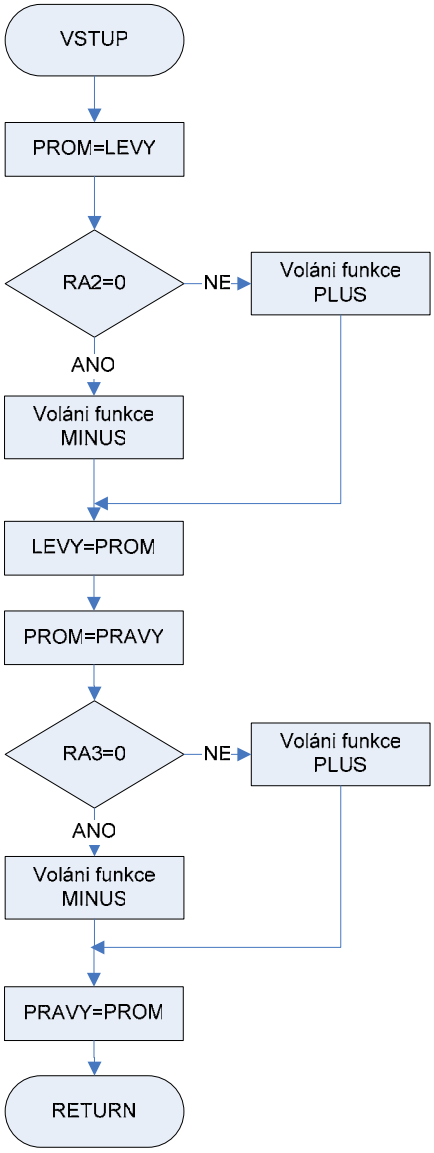
Vývojový diagram funkce ODPPRI



Vývojový diagram funkce ZALOHA



Vývojový diagram funkce VSTUP



Vývojový diagram funkce HODNOT